Sudoku Solver Using image processing

Project Report Fall-sem-21-22

A simple sudoku solver which solves sudoku from image

=

By:

Aileni Rohan Reddy

19BCE2086

Under the supervision of

<u>Viswanathan perumal</u>

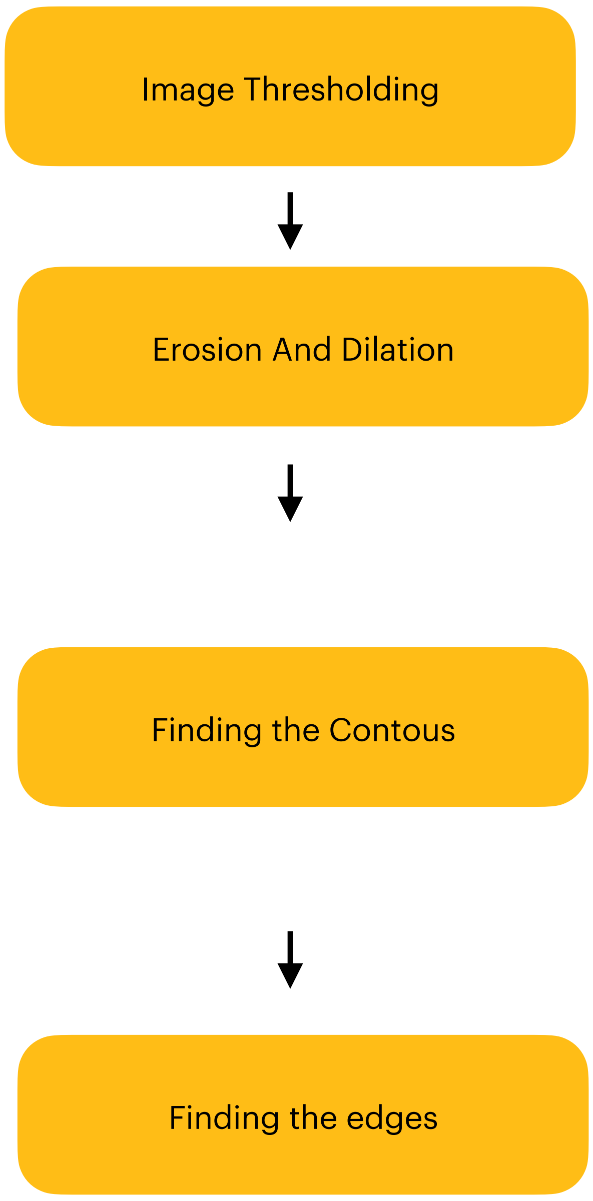
21 November 2021

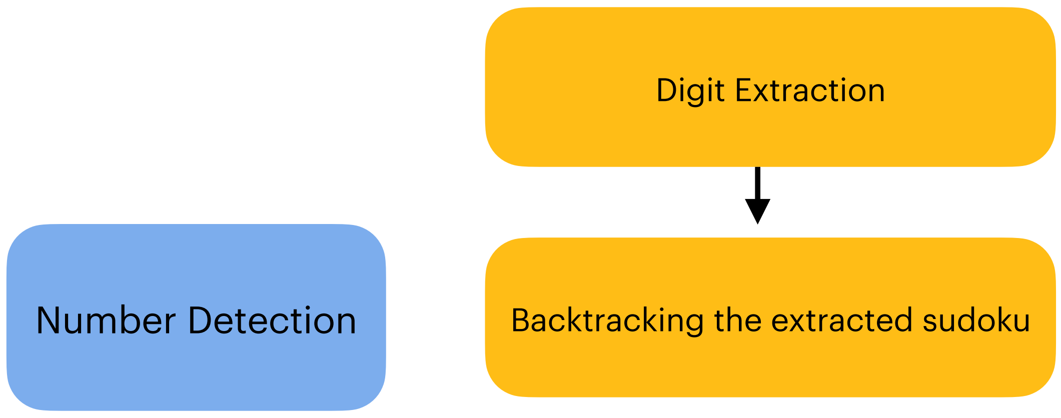
<u>**Existing approach in Our Project Taken as reference in our project:-**</u>

IEEE paper URL:-*https://ieeexplore.ieee.org/document/9033860*

Published March-2019

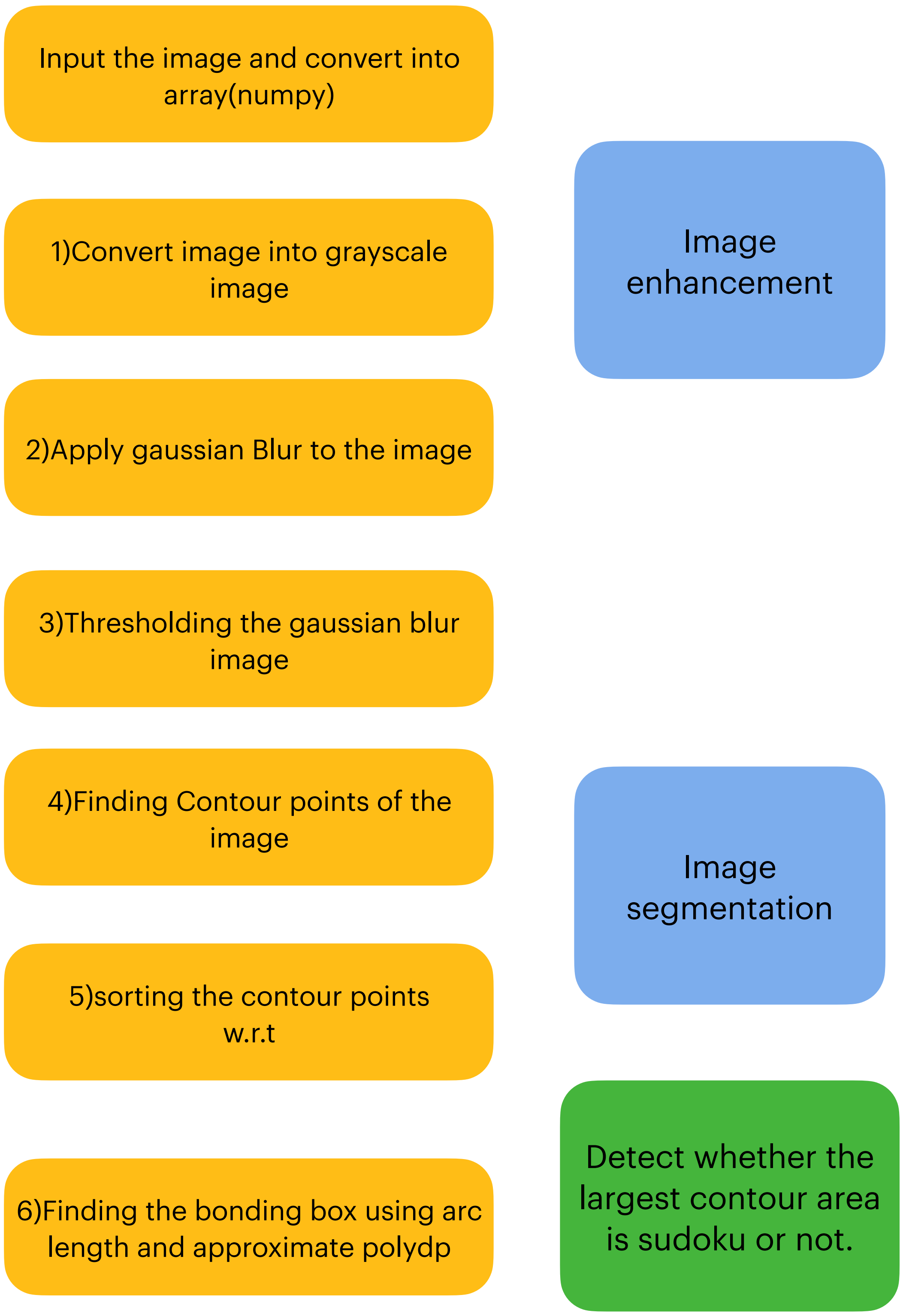
Workflow of Project in the existing approach by the paper

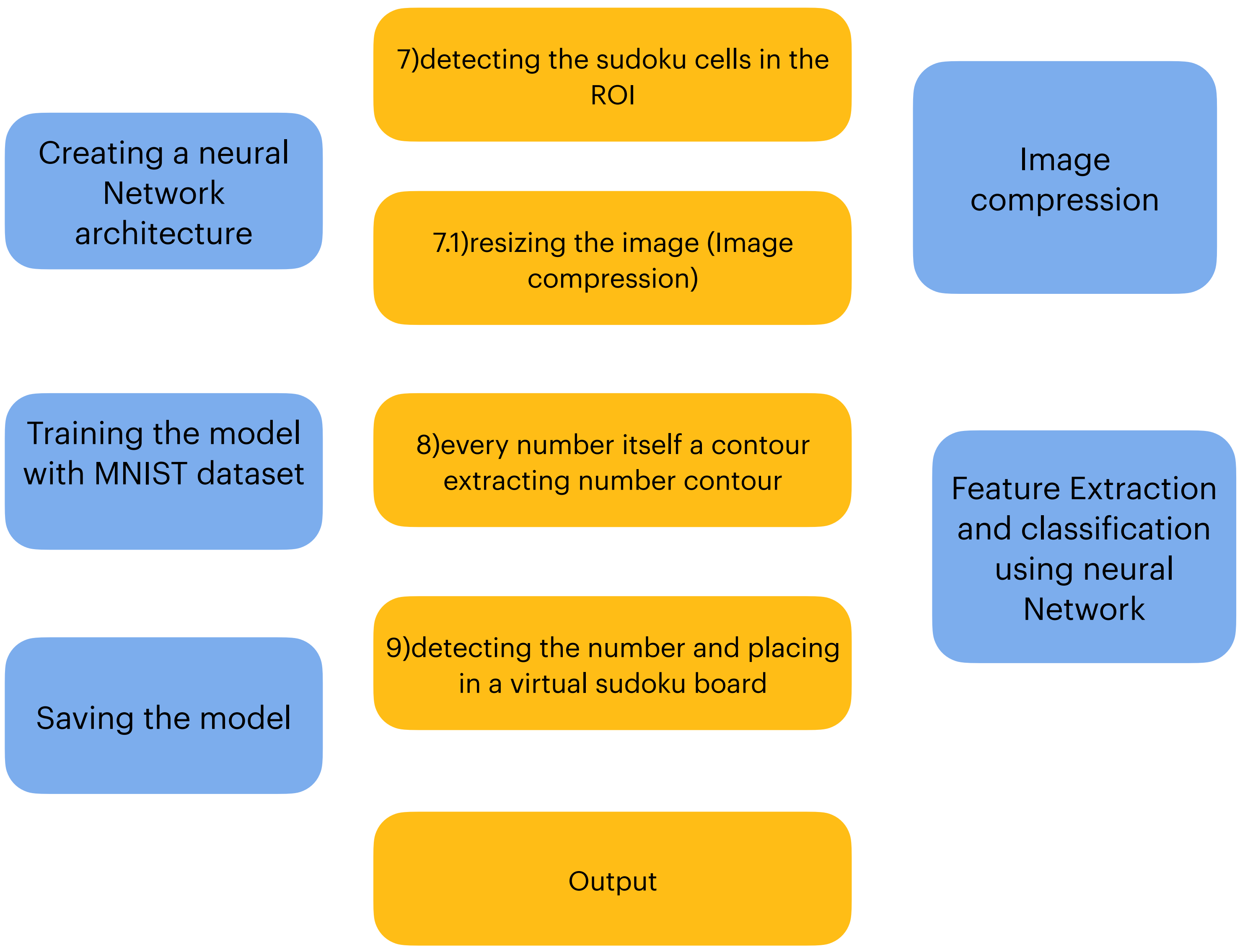



Digit Extraction
Backtracking the extracted sudoku
Number Detection

WORKFLOW OF Our PROJECT

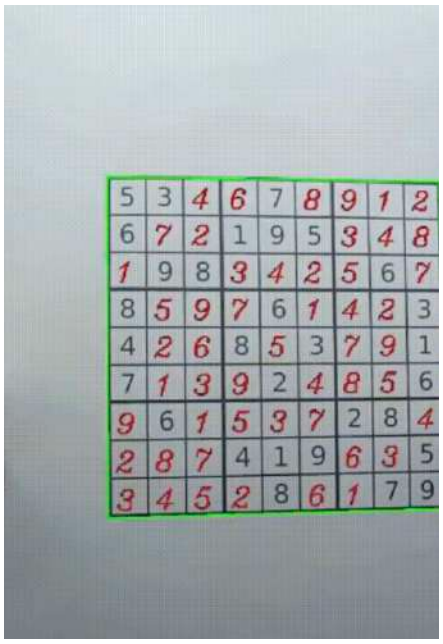
(The above work flow is research based can be changed based on the output of the dataset)


Input the image and convert into array(numpy)
1)Convert image into grayscale image
2)Apply gaussian Blur to the image
3)Thresholding the gaussian blur image
4)Finding Contour points of the image
5)sorting the contour points w.r.t
6)Finding the bonding box using arc length and approximate polydp
Image enhancement
Image segmentation
Detect whether the largest contour area is sudoku or not.


Creating a neural Network architecture
7)detecting the sudoku cells in the ROI
Image compression
7.1)resizing the image (Image compression)
Training the model with MNIST dataset
8)every number itself a contour extracting number contour
Feature Extraction and classification using neural Network
Saving the model
9)detecting the number and placing in a virtual sudoku board
Output

So what is difference between Proposed model and our model:-

1) _In the research paper it is clearly mentioned that the the sudoku On which we are taking picture need to be clean In order to detect the image from the plain paper like

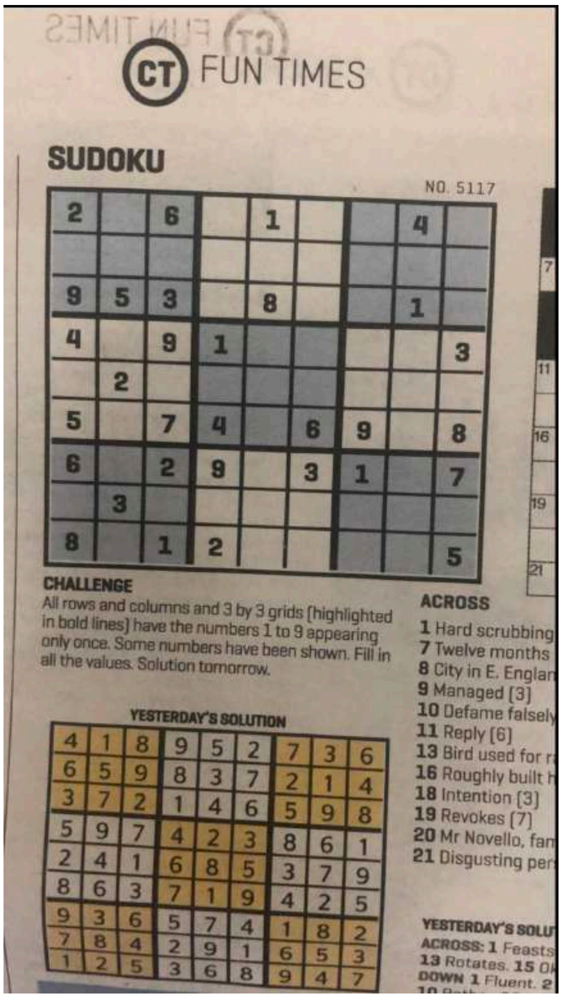


5	3	4	6	7	8	9	1	2
6	7	2	1	9	5	3	4	8
1	9	8	3	4	2	5	6	7
8	5	9	7	6	1	4	2	3
4	2	6	8	5	3	7	9	1
7	1	3	9	2	4	8	5	6
9	6	1	5	3	7	2	8	4
2	8	7	4	1	9	6	3	5
3	4	5	2	8	6	1	7	9

But images won't be this clean If I take an image from the News Paper

<u>This image is taken from the Original News paper this contains Two closing boxed and they both are sudoku</u>

2)Tranformation of Image:-



CT FUN TIMES

SUDOKU

NO. 5117

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

CHALLENGE
All rows and columns and 3 by 3 grids (highlighted in bold lines) have the numbers 1 to 9 appearing only once. Some numbers have been shown. Fill in all the values. Solution tomorrow.

YESTERDAY'S SOLUTION

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

ACROSS
1 Hard scrubbing
7 Twelve months
8 City in E. Englan
9 Managed (3)
10 Defame falsely
11 Reply (6)
13 Bird used for r
16 Roughly built h
18 Intention (3)
19 Revokes (7)
20 Mr Novello, fan
21 Disgusting per

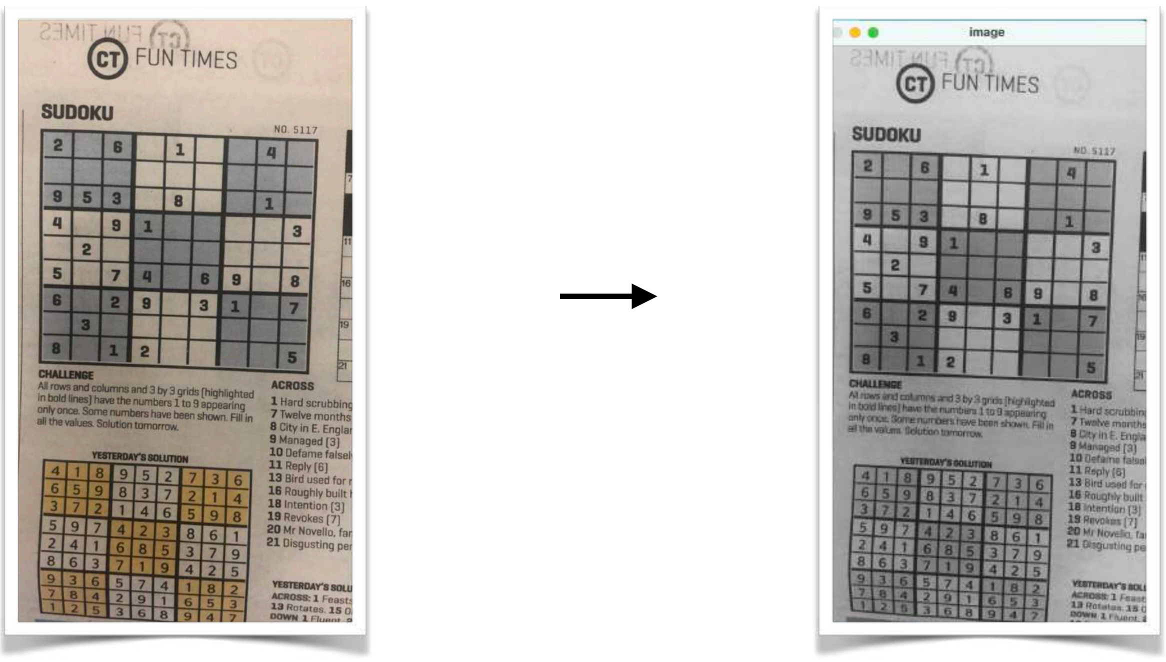
YESTERDAY'S SOLU
ACROSS: 1 Feasts
13 Rotates. 15 O
DOWN 1 Fluent. 2

In the Reference Research paper It is not Discussed about the Transformation of image from one plane to another and more about this topic will be discuss below

<u>Image Enhacement:-</u>

First we have converted the image into To grayscale According to the above research paper HSV seperated the image intensity from chroma or color information .This is used mainly when we use histogram equalization so this process is used to increase contrast of the image but our aim is to to detect the edges of the sudoku box which is not required.(can also use to remove shadows but that is not required .

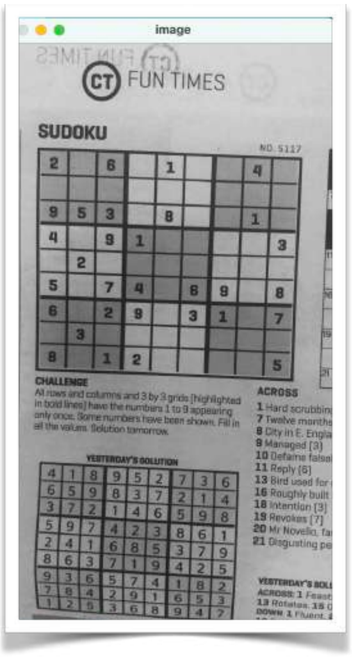
Research paper:-<u>https://www.researchgate.net/publication/258566696_A_Comparative_Study_of_Histogram_Equalization_Based_Image_Enhancement_Techniques_for_Brightness_Preservation_and_Contrast_Enhancement</u>

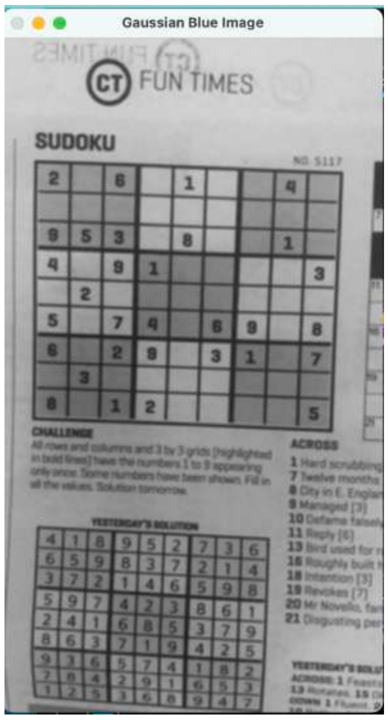


Now I have converted the GrayScale into Gaussian Blur:-

As said above our aim is to detect the edges.so what is an edge? Is it a point in the image there is sudden increase in intensity of the pixel

```
def GaussianBlur(self):
    self.blur=cv2.GaussianBlur(self.gray,(7,7),3)
    return self.blur
```

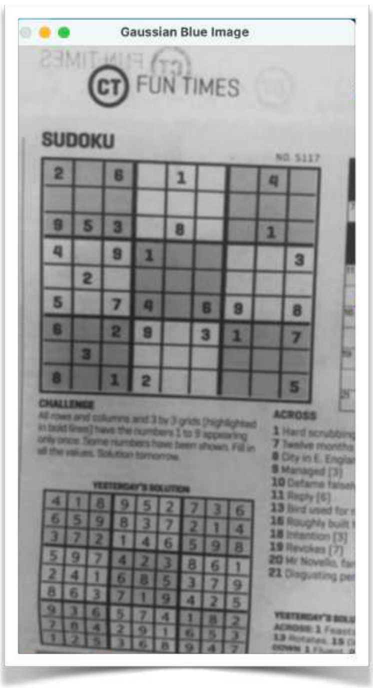




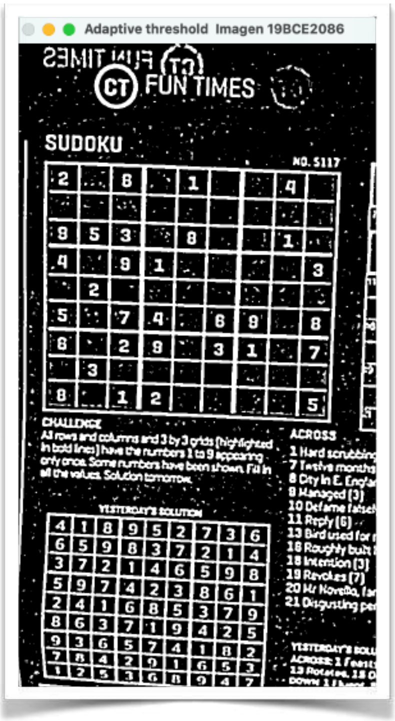
Thresholding the Blur image:-

For my project I use Adaptive Thresholding Because the sudoku is a wide range of colors so there can be many colors in the image One newspaper may have one type of border color and another newspaper may have another type of color so to remove this type of Confusion we use adaptive threshold where threshold value will be the average of the segment of the image .

```
def threshold(self):
self.thresh=cv2.adaptiveThreshold(self.blur,255,cv2.ADAPTIVE_THRESH_MEAN_C,cv2.THRESH_BINARY,11,2)

    self.thresh=cv2.bitwise_not(self.thresh)

    return self.thresh
```



$\longrightarrow$

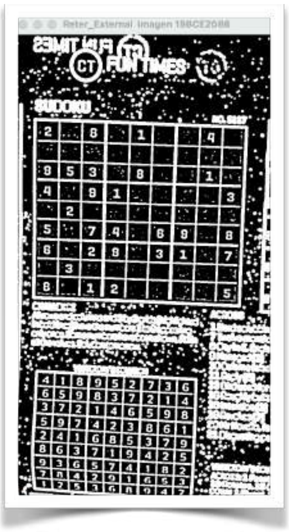


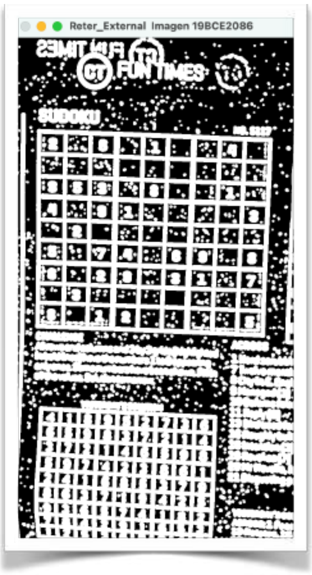
<u>Image Segmentation:-</u>

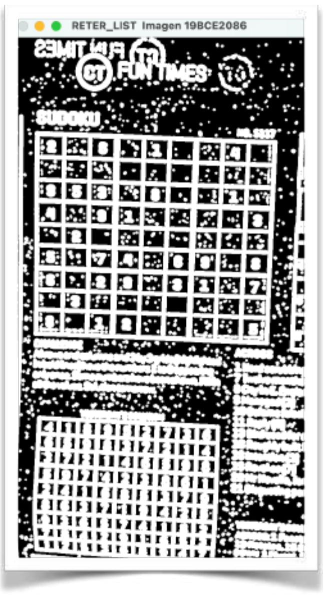
We will use Opencv inbuilt Countour algorithm

```
def contours(self):
    cnts = cv2.findContours(self.thresh.copy(),
cv2.RETR_EXTERNAL, cv2.CHAIN_APPROX_SIMPLE)
    cnts = imutils.grab_contours(cnts)
    cnts = sorted(cnts, key=cv2.contourArea, reverse=True)
    self.cnts=cnts
    return  cnts
```

There are 2 type of Contor algorithm one is RETE_Internal Ans Peter External Wil be hosing 2 algorithm and choosing as Reter_external and best algorithm for our problem







RETER_External   *Reter_Internal*   *Reter_List*

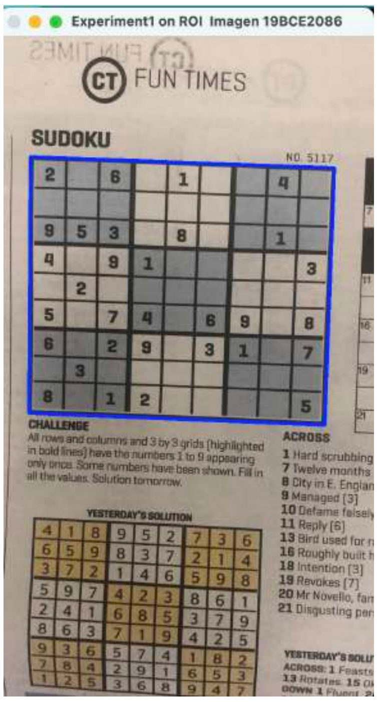
From the above 3 algorithm we chose Reter_Extenal Algorithm as It takes only External Counter algorithms so if there is some Noise Internally it is Unnoticed

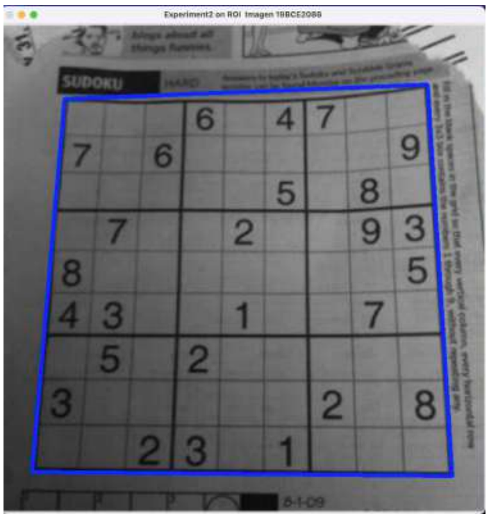
Image Feature Extraction:-

Now after getting the counters we needed to extract the sudoku box so we use approx ploy Dp method with the Help of Arc length Which used to predict where the curve closes or Not

```
def ROIpoints(self):
    self.puzzlecnt=[]
    for c in self.cnts:
        peri = cv2.arcLength(c, True)
        approx = cv2.approxPolyDP(c, 0.02 * peri, True)
        if len(approx) == 4:
            x, y, w, h = cv2.boundingRect(approx)
            ROI = self.image[y:y + h, x:x + w]
            self.puzzlecnt.append(approx)
            break
    return self.puzzlecnt
```

We can see in the above code we have mentioned that the length of counter is greater than 4 so that it can form in any rectangle





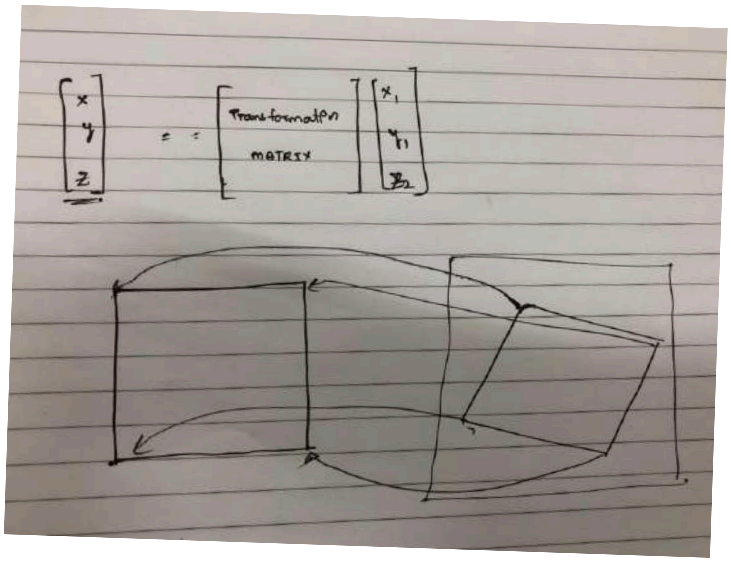
Experiment

No2

Experiment No1

We can see in the experiment no 2 that the image is more irregular shape that the experiment No1

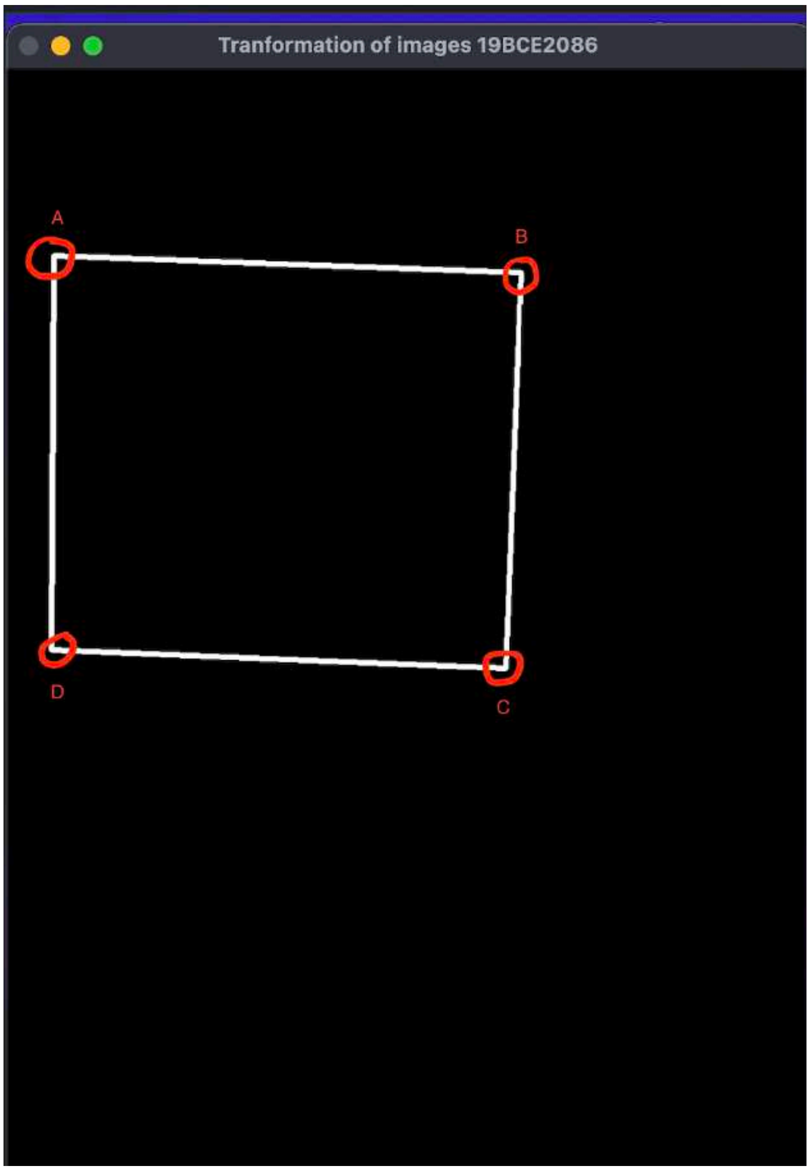
<u>Tranformation of image from One plane to another</u>



Now we can see that the picture in the above Experiments the in different space the make the above rectangle to our plane we use Transformation of coordinates from linear Algebra where we convert point from one plane to another plane

```
def order_points(self,pts):
    rect = np.zeros((4, 2), dtype="float32")
    # the top-left point will have the smallest sum, whereas
    # the bottom-right point will have the largest sum
    s = pts.sum(axis=1)
    rect[0] = pts[np.argmin(s)]
    rect[2] = pts[np.argmax(s)]
    # now, compute the difference between the points, the
    # top-right point will have the smallest difference,
    # whereas the bottom-left will have the largest difference
    diff = np.diff(pts, axis=1)
    rect[1] = pts[np.argmin(diff)]
    rect[3] = pts[np.argmax(diff)]
    # return the ordered coordinates
    return rect
```

```
def tranformation(self):
    image=self.image
    points=np.array(self.puzzlecnt).reshape(4,2)
    rect = self.order_points(points)
    (tl, tr, br, bl) = rect
    widthA = np.sqrt(((br[0] - bl[0]) ** 2) + ((br[1] - bl[1]) ** 2))
    widthB = np.sqrt(((tr[0] - tl[0]) ** 2) + ((tr[1] - tl[1]) ** 2))
    maxWidth = max(int(widthA), int(widthB))
    heightA = np.sqrt(((tr[0] - br[0]) ** 2) + ((tr[1] - br[1]) ** 2))
    heightB = np.sqrt(((tl[0] - bl[0]) ** 2) + ((tl[1] - bl[1]) ** 2))
    maxHeight = max(int(heightA), int(heightB))
    dst = np.array([
        [0, 0],
        [maxWidth - 1, 0],
        [maxWidth - 1, maxHeight - 1],
        [0, maxHeight - 1]], dtype="float32")
    M = cv2.getPerspectiveTransform(rect, dst)
    warped = cv2.warpPerspective(image, M, (maxWidth, maxHeight))
    return warped
```



So if I Draw a Counters in a blank Picture

Now Theoretically

A=min(x+y)

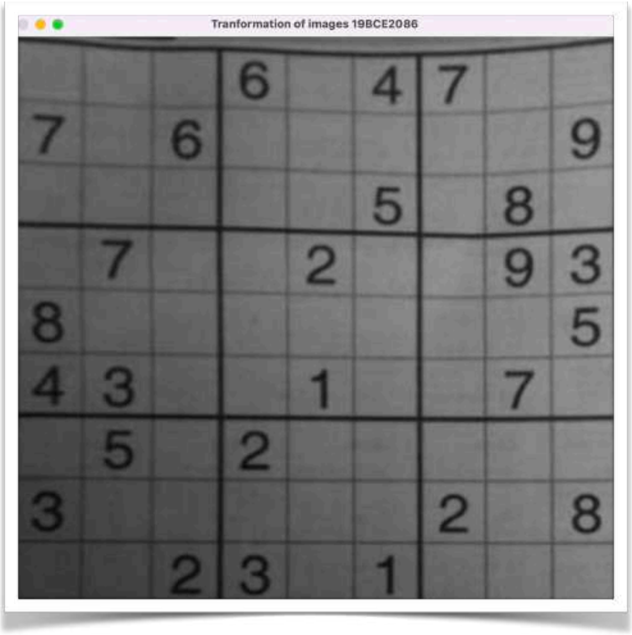
C=max(x+y)

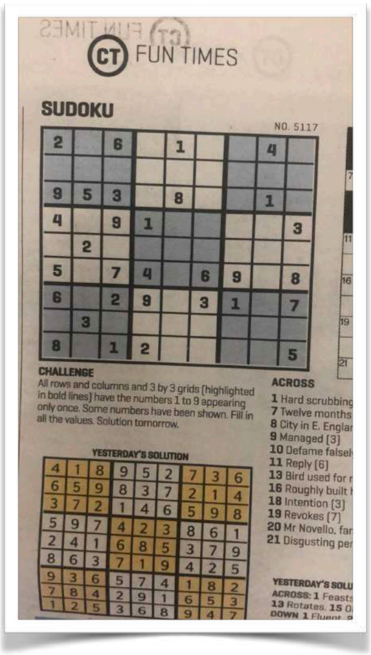
B=min(x-y)

D=max(m+y)

So from this points I can order the end points form the images

$\longrightarrow$




FUN TIMES
SUDOKU
NO. 5117

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

CHALLENGE
All rows and columns and 3 by 3 grids (highlighted in bold lines) have the numbers 1 to 9 appearing only once. Some numbers have been shown. Fill in all the values. Solution tomorrow.

YESTERDAY'S SOLUTION

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

ACROSS
1 Hard scrubbing
7 Twelve months
8 City in E. Englar
9 Managed (3)
10 Defame falsel
11 Reply (6)
13 Bird used for r
16 Roughly built
18 Intention (3)
19 Revokes (7)
20 Mr Novello, far
21 Disgusting per

YESTERDAY'S SOLU
ACROSS: 1 Feasts
13 Rotates. 15 O
DOWN 1 Fluent

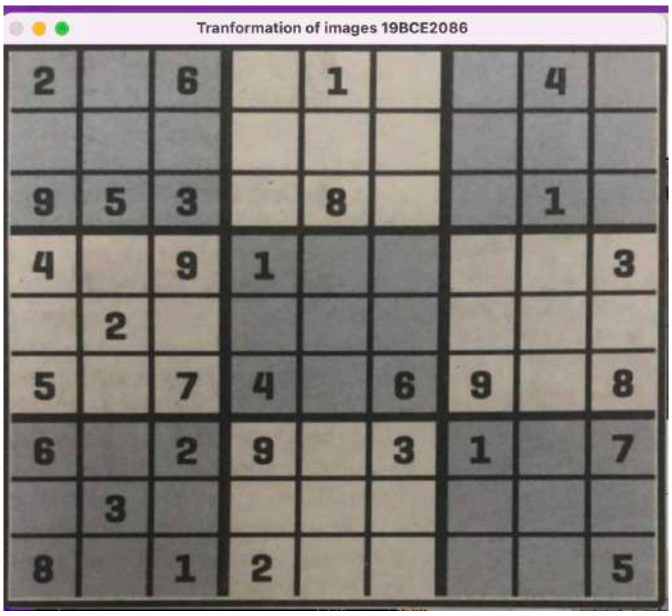
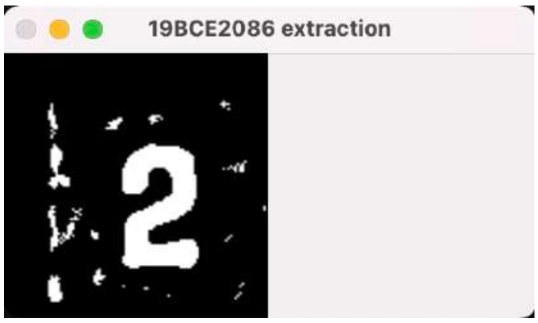
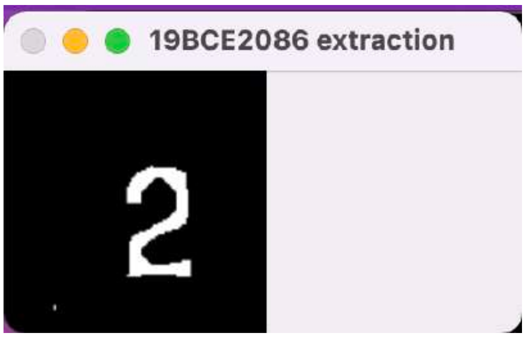


Image Classification

Now the best part Extracting Numbers from the images first after extraction we need to prepare the neural network model .So any CNN model takes binary Images as the input for the detection of numbers so every number has some noise in it





Now we can see the noise is reduce by selecting the Biggest object in the frame

Image compression:-

So why do we even need image compression in my project.Thats because while training the Model I have compressed the image into 224x224 so that more features from the model is extracted but to do we need to also implement our image into 224x224 so that it guesses correct number.

So we will be using DCT(Discrete cosine transform for JPEG)images and Entropy Encoding for all other image compression. As our image is grey image it will be most lossless compression.

According to the cv2 Documentation
INTER_AREA uses Entropy Encoding algorithm for resizing images lossless

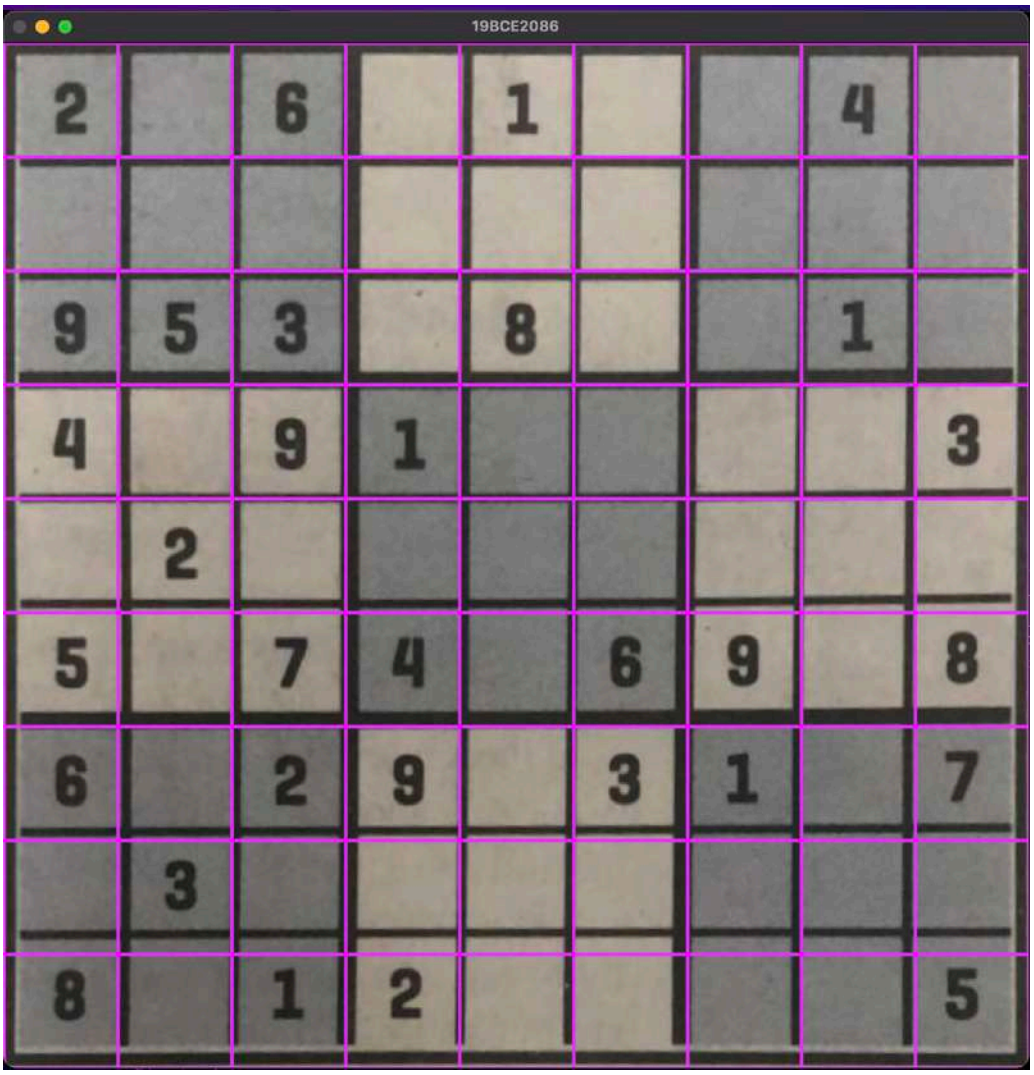
How My model Detect the Each cell in the box ?

So the initiation is to create a rectangle and convert the into 900x900 square so that any image whatever the size maybe it will be converted into 900x900 image .so there are 9 rows and 9 columns in sudoku so each cell is of size is 100x100

Code :-

```
image=cv2.resize(image,
(900,900),interpolation=cv2.INTER_AREA)
```

So now after compressing the image then we draw the lines to throw some clarity on it



DETECTING NUMBERS IN THE BOX:-

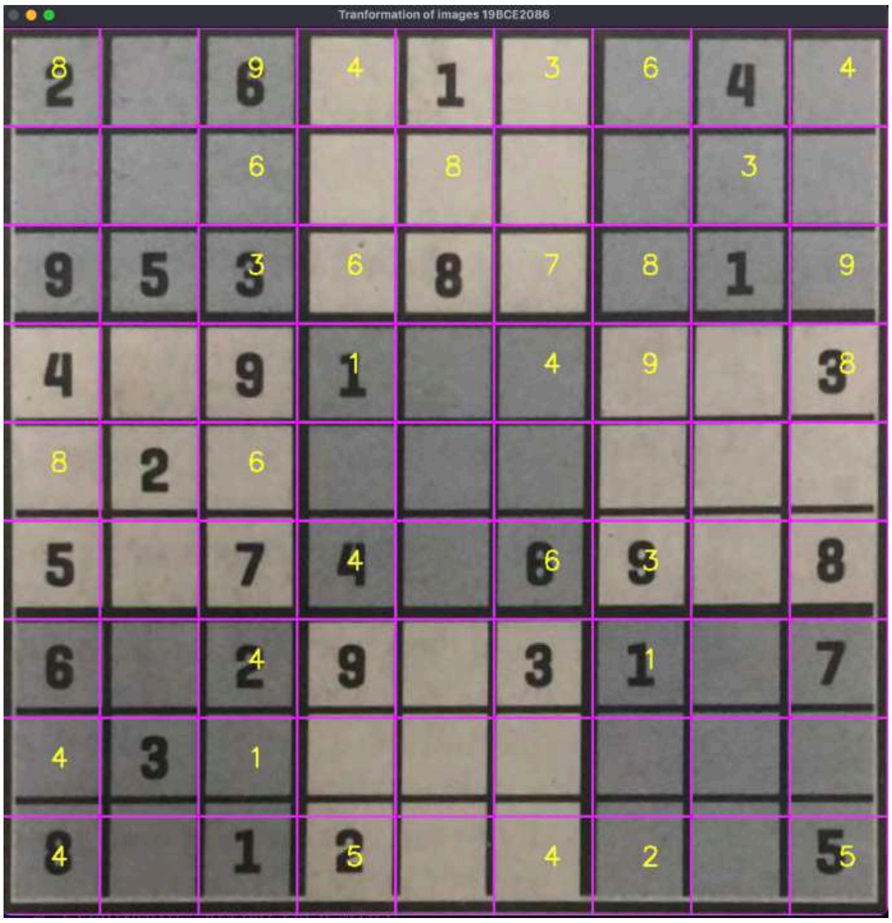
So to detect the number from the each cell we needed to Create a can Architecture to detect the numbers before looking into CNN architecture we will solver

So we will be training for the data set from website:-

http://www.ee.surrey.ac.uk/CVSSP/demos/chars74k/

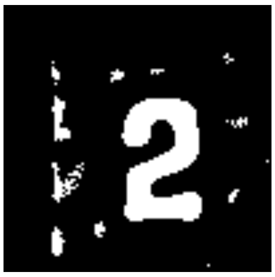
Why not MNIST?

Firstly I trained with Mnist Dataset but the output Is a follows by using mnist

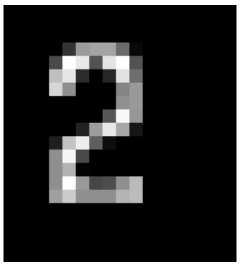


From the above image we can see that the cells can't detect the numbers exactly

And Also in the cells where there are not numbers there is an number detected it So to remove this we will looking at each cell



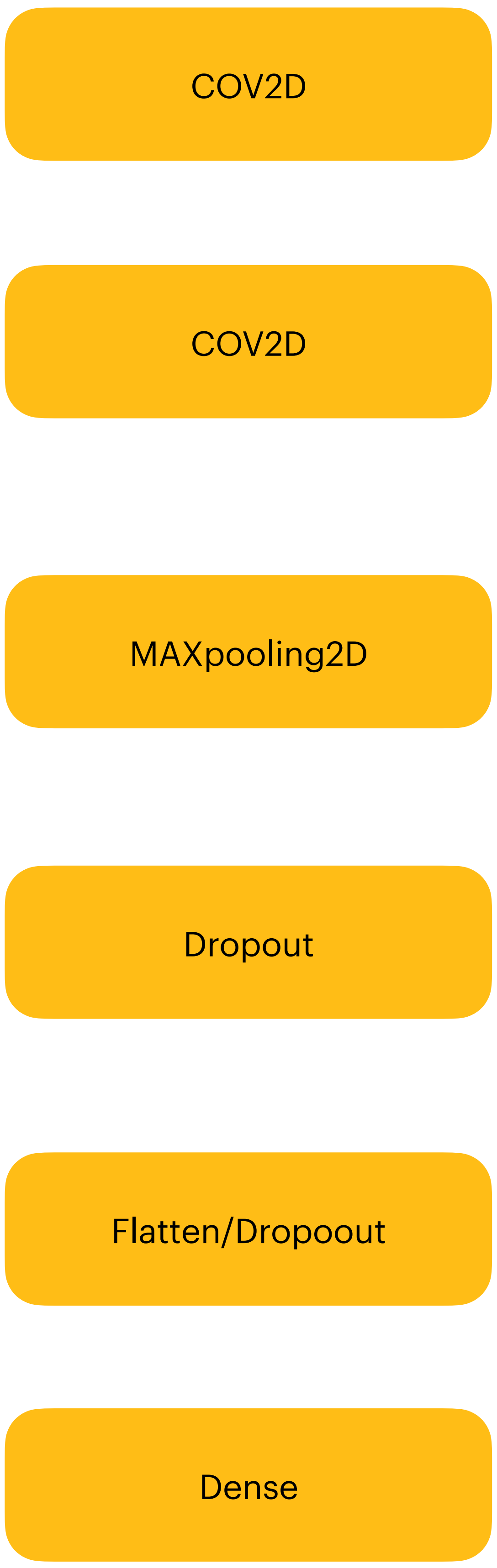
There is lot of noice in each cell to remove it we will use contours to remove it



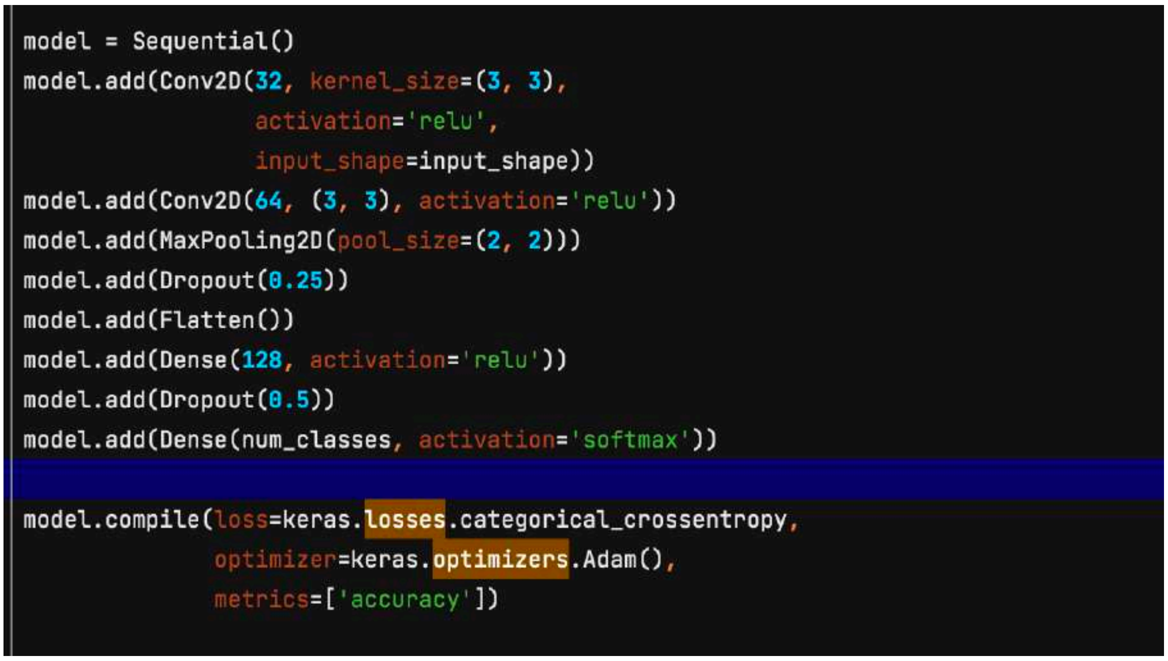
This is how the image looks after using counters

But the minis Is created for handwriting detection of numbers to remove so thats the reason why we will be using CHAR74 dataset which contains all the data/Images in italic and Bold format

CNN Architecture:-



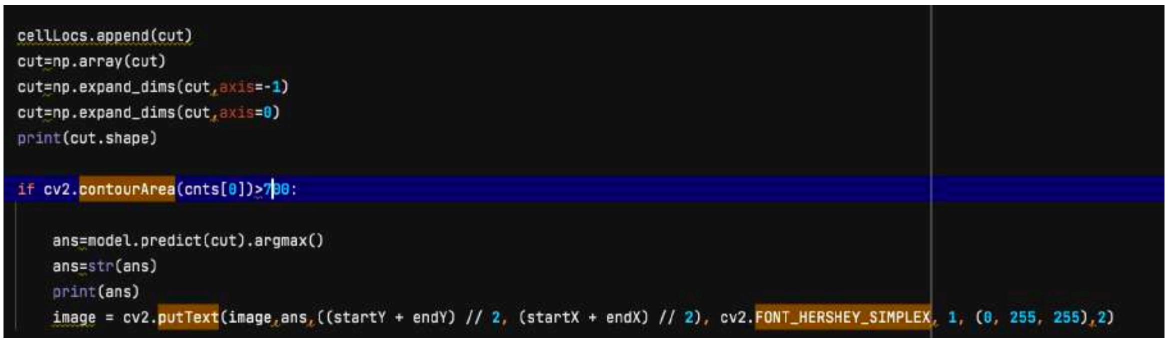
<u>A Dropout layer is used to reduce over fitting the dataset and is set to around 20%</u>



```
model = Sequential()
model.add(Conv2D(32, kernel_size=(3, 3),
                 activation='relu',
                 input_shape=input_shape))
model.add(Conv2D(64, (3, 3), activation='relu'))
model.add(MaxPooling2D(pool_size=(2, 2)))
model.add(Dropout(0.25))
model.add(Flatten())
model.add(Dense(128, activation='relu'))
model.add(Dropout(0.5))
model.add(Dense(num_classes, activation='softmax'))

model.compile(loss=keras.losses.categorical_crossentropy,
              optimizer=keras.optimizers.Adam(),
              metrics=['accuracy'])
```

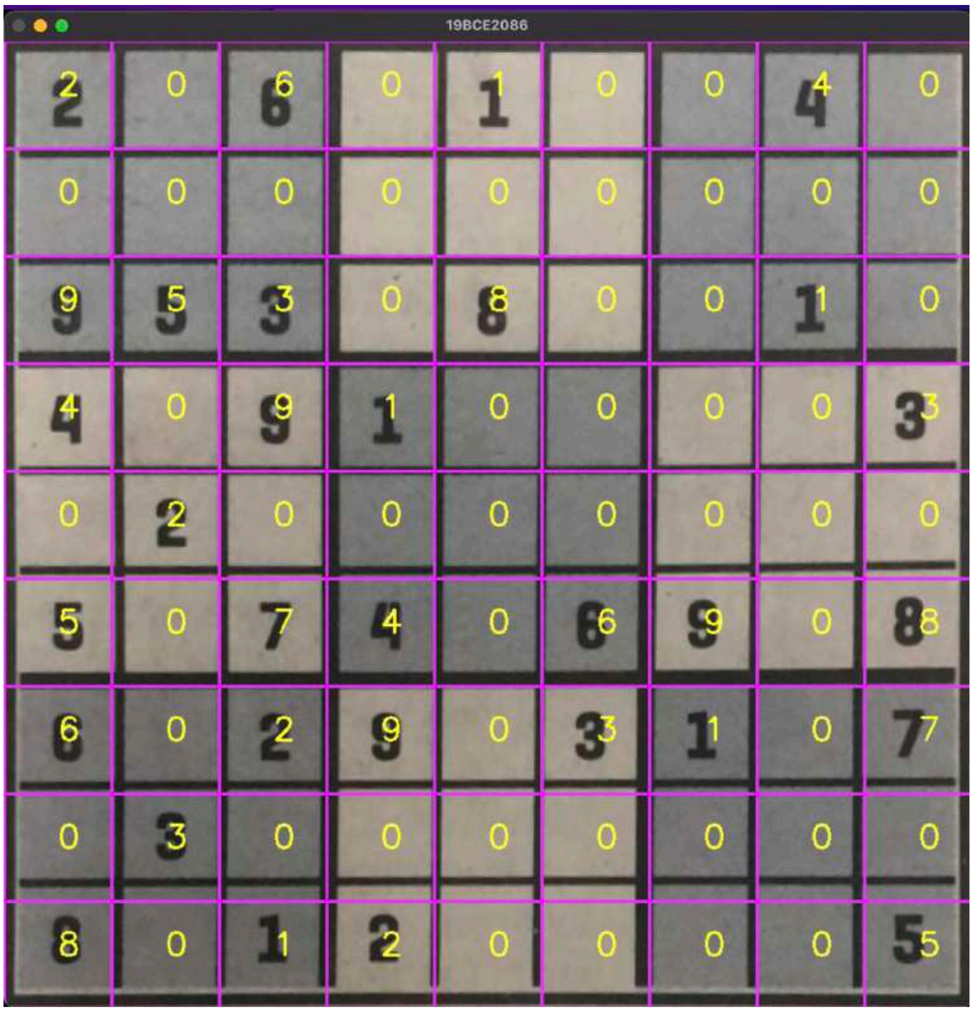
<u>Accuracy for Test Data:-99.234%</u>

<u>To tackle Empty cell we have used CounterArea is set to around 700</u>

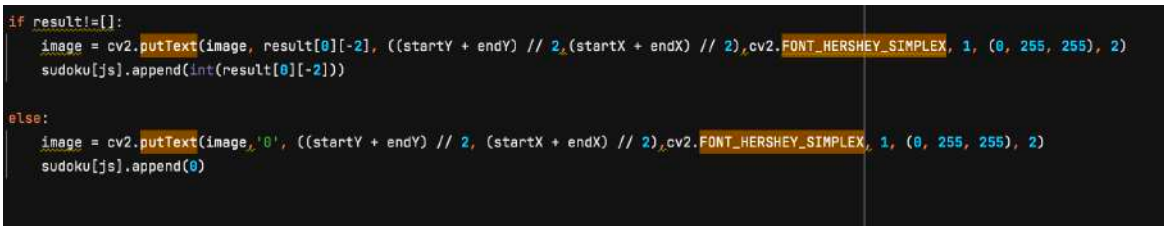


```
cellLocs.append(cut)
cut=np.array(cut)
cut=np.expand_dims(cut,axis=-1)
cut=np.expand_dims(cut,axis=0)
print(cut.shape)

if cv2.contourArea(cnts[0])>700:

    ans=model.predict(cut).argmax()
    ans=str(ans)
    print(ans)
    image = cv2.putText(image,ans,((startY + endY) // 2, (startX + endX) // 2), cv2.FONT_HERSHEY_SIMPLEX, 1, (0, 255, 255),2)
```

<u>Now using the new model on each cell</u>



<u>Afte</u>r Taking each cell counter area the if the area is less than 700 then the counter cell is labeled as zero I



```
if result!=[]:
    image = cv2.putText(image, result[0][-2], ((startY + endY) // 2,(startX + endX) // 2),cv2.FONT_HERSHEY_SIMPLEX, 1, (0, 255, 255), 2)
    sudoku[js].append(int(result[0][-2]))

else:
    image = cv2.putText(image,'0', ((startY + endY) // 2, (startX + endX) // 2),cv2.FONT_HERSHEY_SIMPLEX, 1, (0, 255, 255), 2)
    sudoku[js].append(0)
```

<u>SOLVING THE SUDOKU:-</u>

After getting the numbers we ill store each cell Detected in an array and will be given to solve_sudoku function:-

We will be using standard backtracking Mechanism to Detect boxes and underline the Sudoku .

Code:-

```
class sol:
    def find_empty(self,bo):
        for i in range(len(bo)):
            for j in range(len(bo[0])):
                if bo[i][j] == 0:
                    return (i, j)  # row, col
        return None
    def solve(self,bo):
        find = self.find_empty(bo)
        if not find:
            return True
        else:
            row, col = find
        for i in range(1,10):
            if self.valid(bo, i, (row, col)):
                bo[row][col] = i
                if self.solve(bo):
                    return True
                bo[row][col] =
        return False
    def valid(self,bo, num, pos):
        # Check row
        for i in range(len(bo[0])):
            if bo[pos[0]][i] == num and pos[1] != i:
                return False
        # Check column
        for i in range(len(bo)):
            if bo[i][pos[1]] == num and pos[0] != i:
                return False
        # Check box
        box_x = pos[1] // 3
        box_y = pos[0] // 3
        for i in range(box_y*3, box_y*3 + 3):
            for j in range(box_x * 3, box_x*3 + 3):
                if bo[i][j] == num and (i,j) != pos:
                    return False
        return True
    def print_board(self,bo):
        for i in range(len(bo)):
            if i % 3 == 0 and i != 0:
                print("- - - - - - - - - - - - - ")
            for j in range(len(bo[0])):
                if j % 3 == 0 and j != 0:
                    print(" | ", end="")
                if j == 8:
                    print(bo[i][j])
                else:
                    print(str(bo[i][j]) + " ", end="")
```

We will just use PrintFunction to print it on the console to Get some clarity

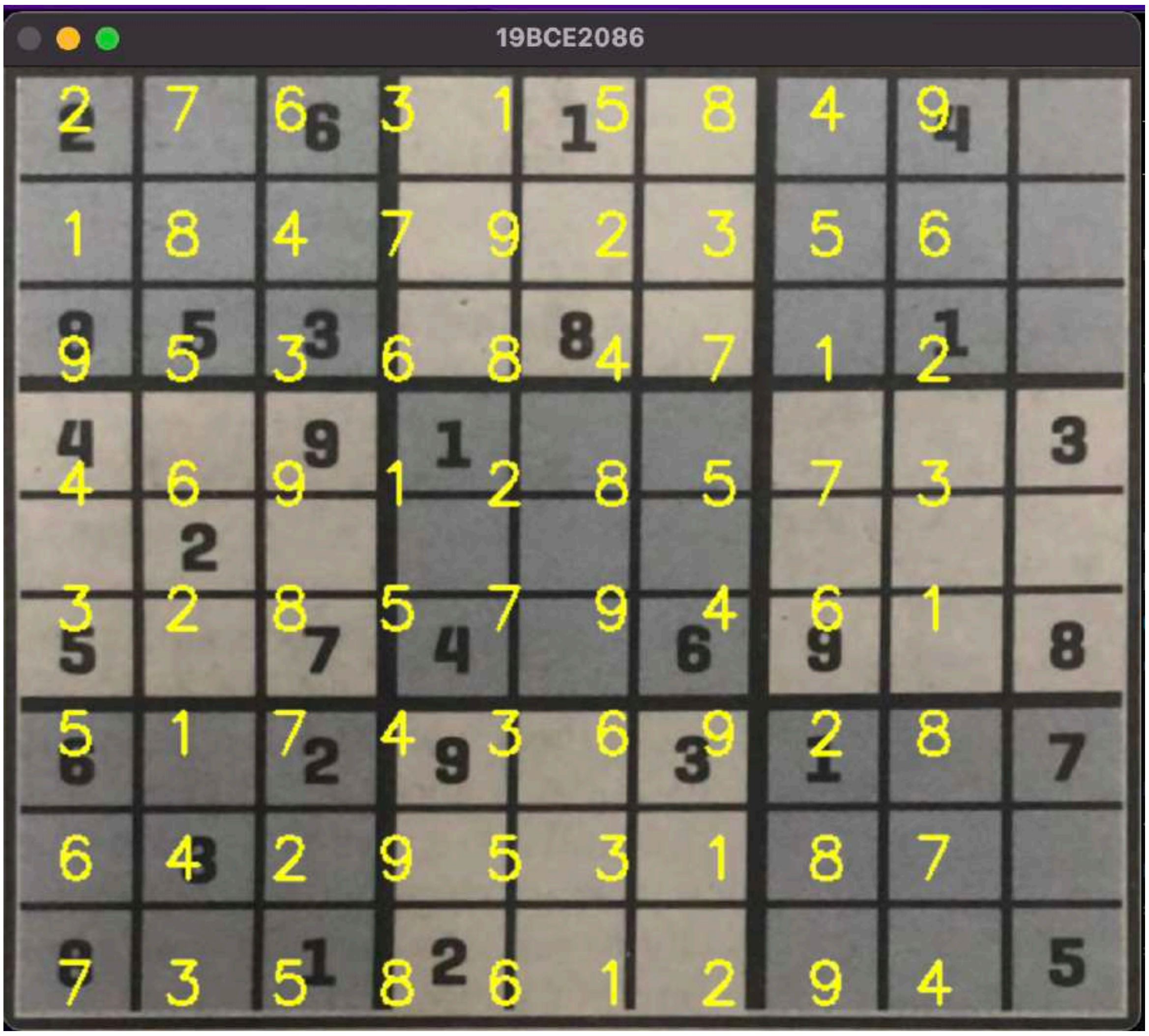
Now after solving sudo from the array the Next thing is to write the sudoku from the array to image:

Code:-

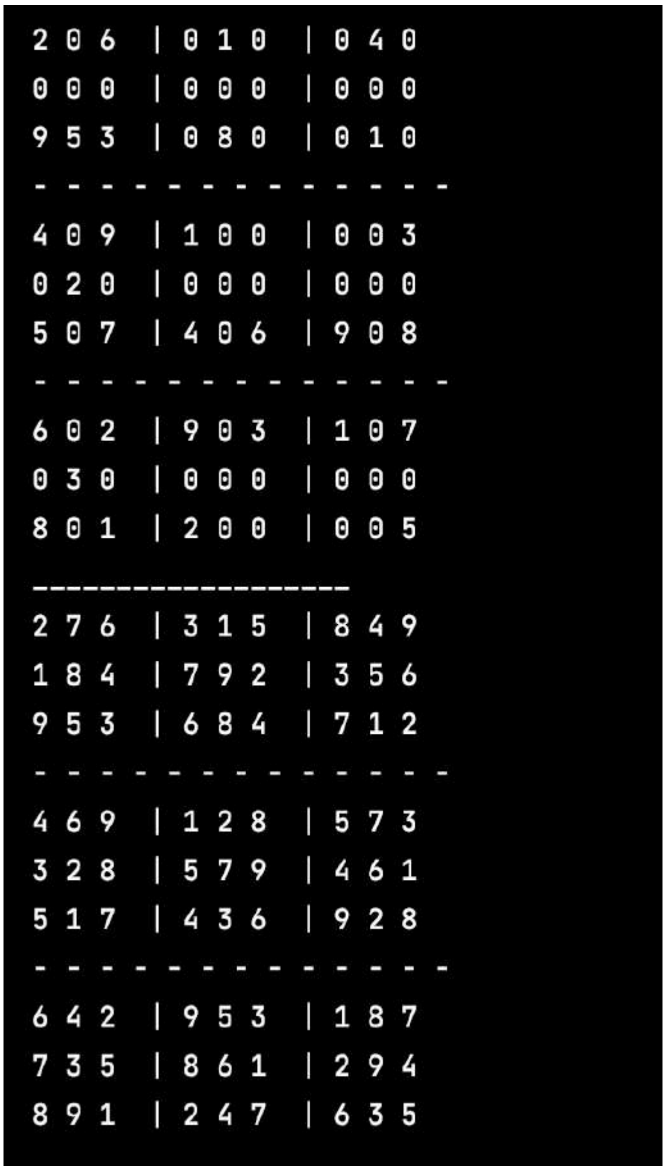
```
def write_on_board(self,image,sudoku):
    stepX = image.shape[1] // 9
    stepY = image.shape[0] // 9
    for y in range(0, 9):
        # initialize the current list of cell locations
        row = []
        js=0
        for x in range(0, 9):
            # compute the starting and ending (x, y)-
coordinates of the
            # current cell
            startX = x * stepX
            startY = y * stepY
            endX = (x + 1) * stepX
            endY = (y + 1) * stepY
            image = cv2.putText(image, str(sudoku[x][y]),
((startY + endY) // 2, (startX + endX) //
2),cv2.FONT_HERSHEY_SIMPLEX, 1, (0, 255, 255), 2)
    return image
```

The Above Function will take the image and sudoku where image is a open image object and sudoku is just and array and will take image as 900x900 clean image

Output:-



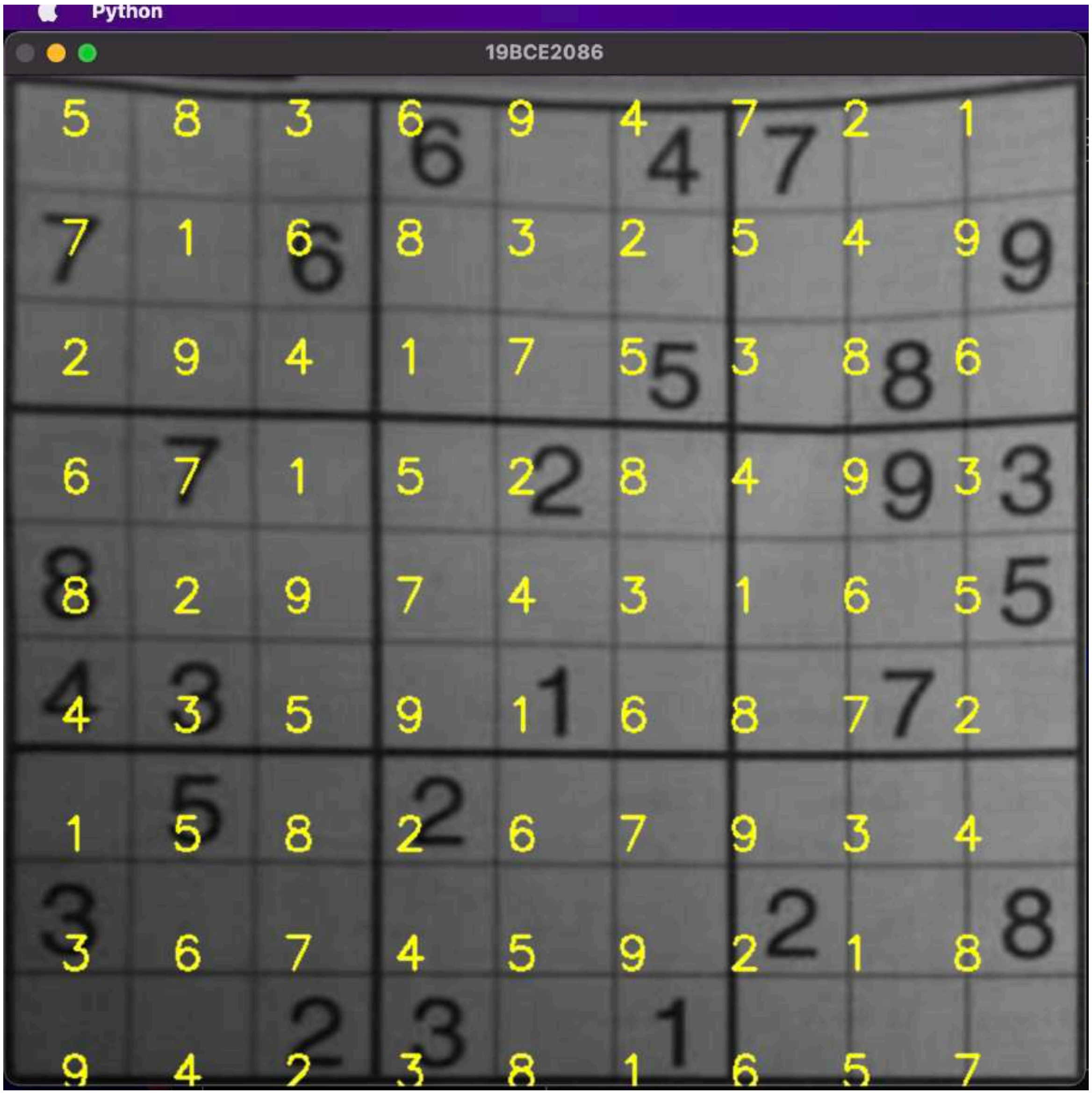
Output in the console:-



```
2 0 6  | 0 1 0  | 0 4 0
0 0 0  | 0 0 0  | 0 0 0
9 5 3  | 0 8 0  | 0 1 0
- - - - - - - - - - - - -
4 0 9  | 1 0 0  | 0 0 3
0 2 0  | 0 0 0  | 0 0 0
5 0 7  | 4 0 6  | 9 0 8
- - - - - - - - - - - - -
6 0 2  | 9 0 3  | 1 0 7
0 3 0  | 0 0 0  | 0 0 0
8 0 1  | 2 0 0  | 0 0 5
___________________
2 7 6  | 3 1 5  | 8 4 9
1 8 4  | 7 9 2  | 3 5 6
9 5 3  | 6 8 4  | 7 1 2
- - - - - - - - - - - - -
4 6 9  | 1 2 8  | 5 7 3
3 2 8  | 5 7 9  | 4 6 1
5 1 7  | 4 3 6  | 9 2 8
- - - - - - - - - - - - -
6 4 2  | 9 5 3  | 1 8 7
7 3 5  | 8 6 1  | 2 9 4
8 9 1  | 2 4 7  | 6 3 5
```

ABOUT THE ACCURACY:-

TEST ON IMAGE 2:-



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

Similarly around the 25 sudoku papers have been solved using this model and all the sudoku are from news papers which have some noise in which around 24 pictures showed 100% accuracy in the predicting the model

Accuracy of My sudoku solver Model is:-97% which is far Better than 90% by the research paper the I referred to do so

REFEEENCES:-

- https://ieeexplore.ieee.org/document/9033860
- http://www.journaleca.com/gallery/jeca%20-%202306.pdf
- https://analyticsindiamag.com/solve-sudoku-puzzle-using-deep-learning-opencv-and-backtracking/
- https://www.irjet.net/archives/V7/i10/IRJET-V7I10253.pdf(MAIN REFERENCE)
-
- https://project-archive.inf.ed.ac.uk/ug4/20201867/ug4_proj.pdf
- https://ijritcc.org/index.php/ijritcc/article/view/4057
- https://www.researchgate.net/publication/333418678_Image_Detection_and_Digit_Recognition_to_solve_Sudoku_as_a_Constraint_Satisfaction_Problem