Exploratory Project

Under the Guidance of
Prof. V. N. Mishra

Team members:
Komal Garg (20095054)
Shashi Kant (2009106)
Electronics Engineering
(B.Tech 4th Semester)

REAL TIME SUDOKU SOLVER

About this project

Many people try solving Sudoku puzzles everyday. These puzzles are usually found in newspapers, magazines and so on. Whenever a person is unable to solve a puzzle or is running short on time to solve the puzzle, it will be very convenient to show the solved puzzle as an augmented reality.

The capabilities of the project include being able to detect and recognize such a puzzle by getting feed from the rear camera of a device. It should be able to solve the game in real time and fill the empty squares with the correct digits, displaying them over the feed from the camera on the screen in the corresponding positions.

Contents

Introduction

The Sudoku Puzzle

Sudoku is a logic-based combinatorial puzzle. When introduced in Japan, it receives its current name Sudoku, meaning” single number” in Japanese. It is now popular all over the world. The game consists of a 9x9 grid divided into 9 square sub-grids (of size 3x3). Some of the squares in the grid contain a number from 1 to 9. The player is presented with a partially filled grid and their aim is to fill the rest of the squares with numbers from 1 to 9. The rules are fairly simple: each row, column and block must contain each of these numbers exactly once

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

Main language and libraries used

PYTHON ***: An interpreted high level general purpose programming language.***

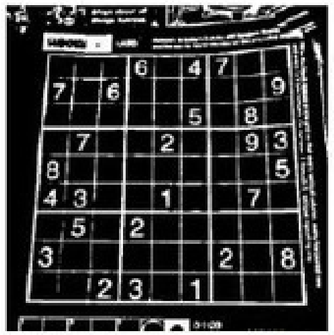
TENSORFLOW ***: An open source library to create neural networks easily.***

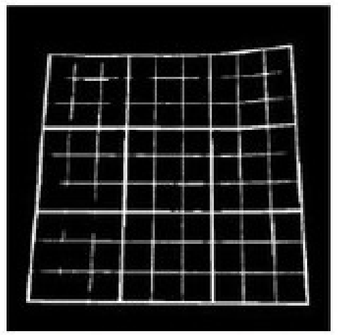
KERAS ***: An interface to the tensorflow library***

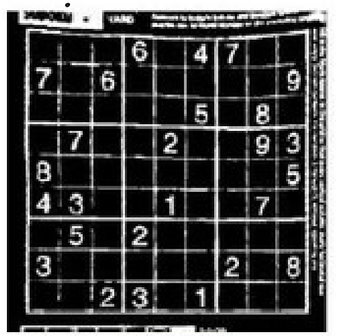
OPENCV ***: A python library used to develop real time computer vision applications***

NUMPY ***: A python library used to work with arrays efficiently***

Flow of code







Capture image from camera

Binarize with adaptive thresholding

Extract largest connected component (CC)

Warp perspective using 4 corners of largest CC

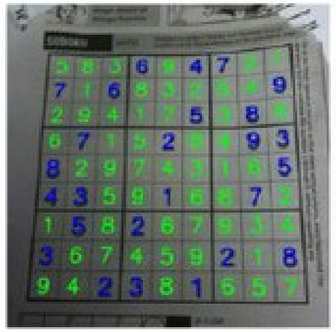
Estimate grid cell centers using structuring elements

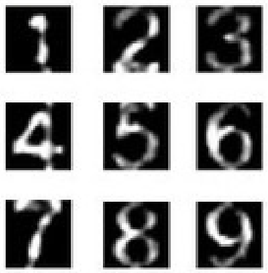
Digit extraction using largest CC within cell

Digit recognition by finding template of least square error

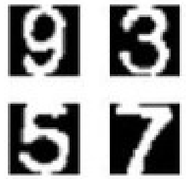
Solve Sudoku using backtracing

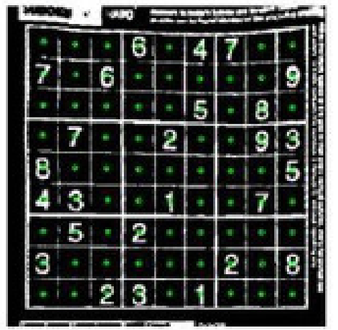
Overlay solution values at cell centers and undo warp perspective





Templates
(class means of training images)





Training the model

1. In the first step, we load the MNIST dataset. We also import all the libraries required such as OpenCV, NumPy, TensorFlow, Keras etc.

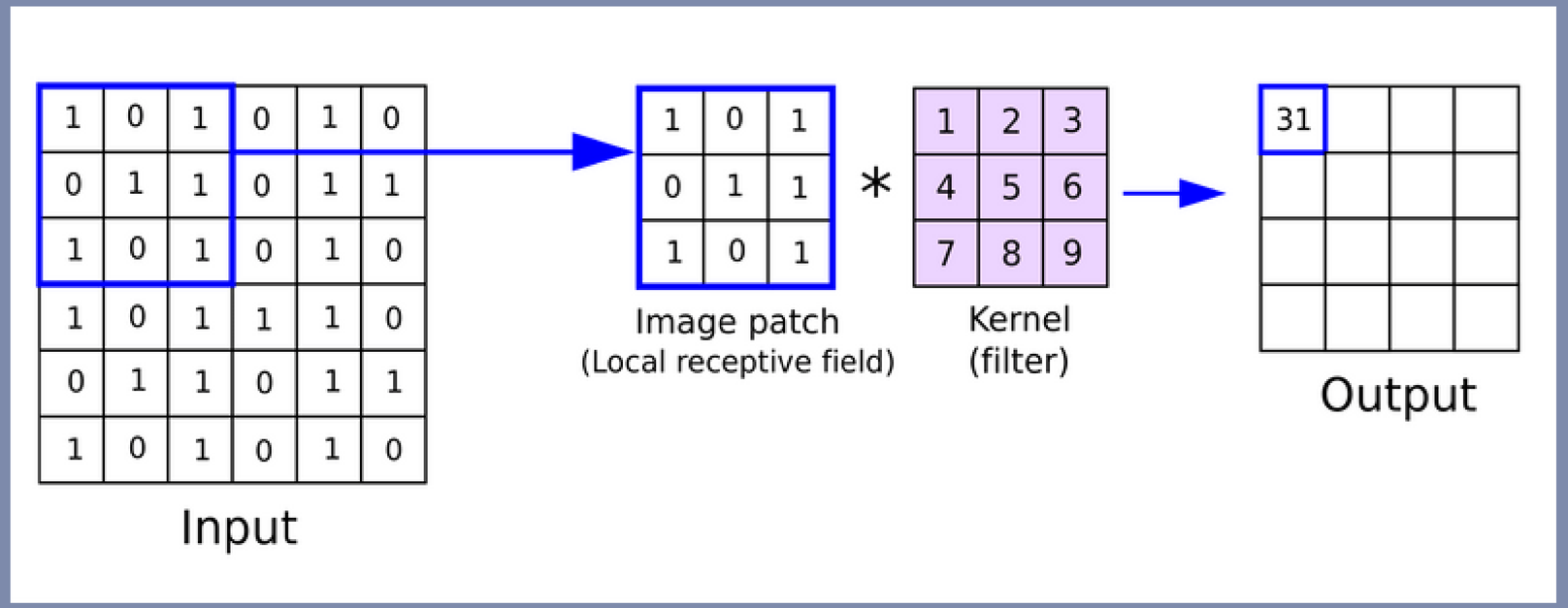
2. Now, we add various layers such as Conv2D, MaxPooling2D, Flatten, Dropout and Dense to our Sequential CNN model. In the last Dense layer, we use the 'SoftMax' activation function to output a vector that gives the probability of each of the 10 classes (i.e., digits from 0-9). Here we use the 'adam' optimizer and 'categorical_crossentropy' as our loss function.

3. This step is the main step where we fit our images in the training set and the test set to our Sequential model that we built using keras library. We have trained the model for 10 epochs (iterations). However, we can train for greater number of epochs to attain higher accuracy lest there occurs over-fitting. We save our model so that we can easily load it for predictions.

Different layers of CNN:

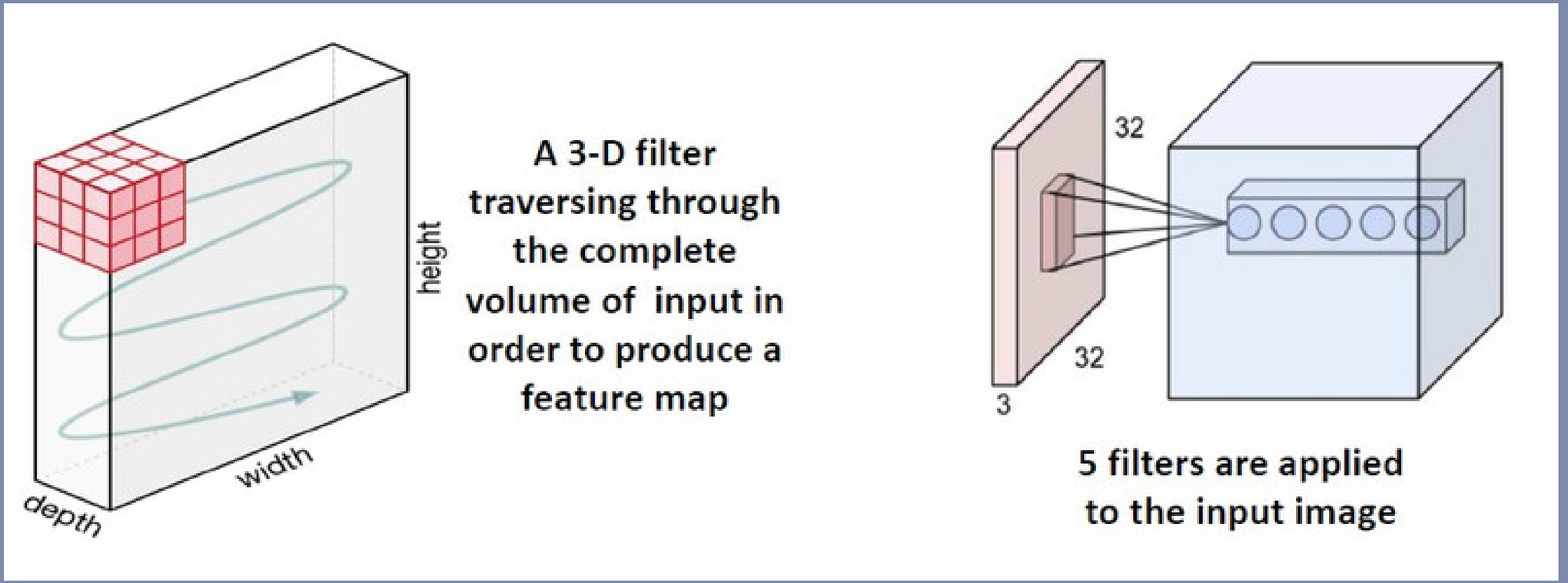
1. The Convolution layer

The convolutional layer is the core building block of a CNN.The layer's parameters consist of a set of learnable filters (or kernels). These filters have a small receptive field, but extend through the full depth of the input volume.



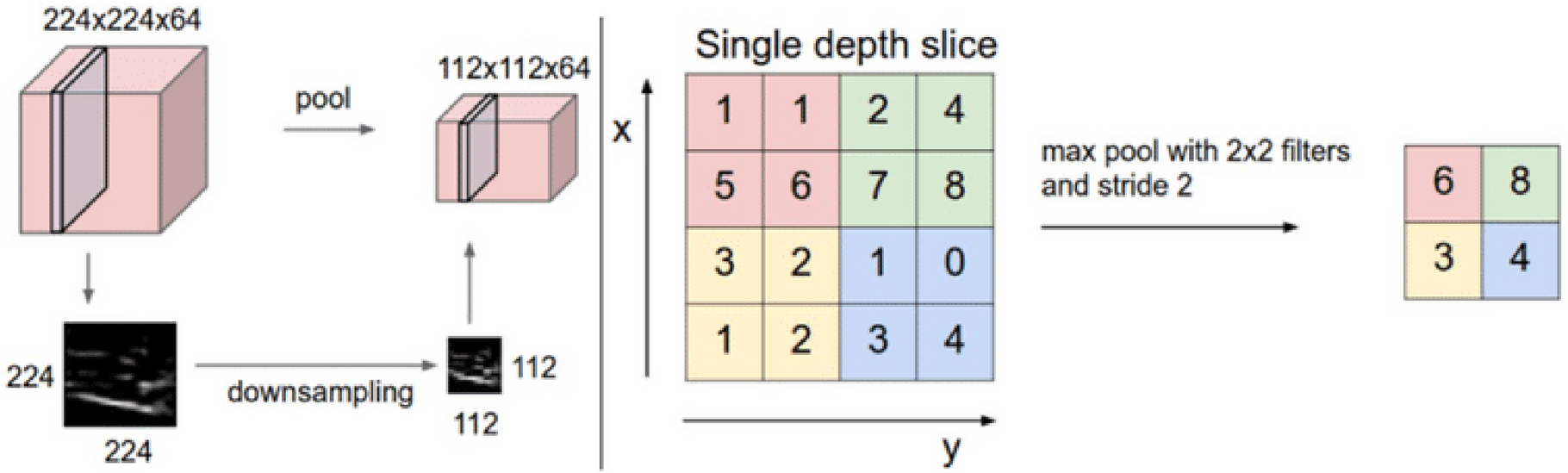
- ***Filters are actually 3 dimensional , one filter produces one 2 D feature map.***

- ***Thus N filters will give rise to N 2 D feature maps which when stacked form a 3 D volume.***



2. Pooling Layer

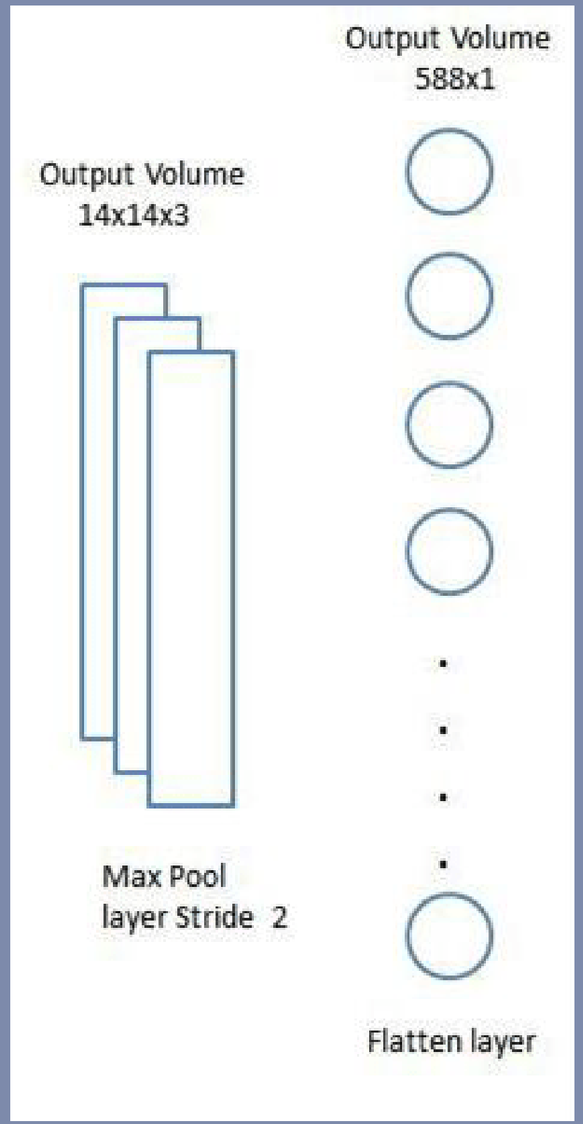
Pooling is done to to progressively reduce the spatial size of the representation to reduce the amount of parameters and computation in the network.



Pooling layer downsamples the volume spatially, independently in each depth slice of the input volume. **Left:** In this example, the input volume of size [224x224x64] is pooled with filter size 2, stride 2 into output volume of size [112x112x64]. Notice that the volume depth is preserved. **Right:** The most common downsampling operation is max, giving rise to **max pooling**, here shown with a stride of 2. That is, each max is taken over 4 numbers (little 2x2 square).

3. The Flatten Layer

- **Converts the data into one dimensional array which is given as input to the next layer.**
- **We flatten the output of long feature vector obtained from pooling.**
- **It is connected to the final classification model, which is called a fully connected layer.**
- **In other words, we put all the pixel data in one line and make connections with the final layer.**



Activation Function:

We have used SoftMax activation function for the output layer and ReLU activation function for other layers.

ReLU: ***The Rectified Linear Unit is the most commonly used activation function in deep learning models. The function returns 0 if it receives any negative input, but for any positive value x it returns that value back. So, it can be written as:***

$$f(x)=max(0,x)$$

Softmax ***is a mathematical function that converts a vector of numbers into a vector of probabilities, where the probabilities of each value are proportional to the relative scale of each value in the vector.***

$$\sigma(\vec{z})_i = \frac{e^{z_i}}{\sum_{j=1}^{K} e^{z_j}}$$

Compiling the model

After building the model, we compile the model and define the loss function and optimizer function. In this model, we use the ‘Adam’ Optimizer and the ‘Categorical Cross Entropy’ as the Loss function for training purpose.

Adam Optimizer:

Adam can be looked at as a combination of RMSprop and Stochastic Gradient Descent with momentum. It uses the squared gradients to scale the learning rate like RMSprop and it takes advantage of momentum by using moving average of the gradient instead of gradient itself like SGD with momentum.

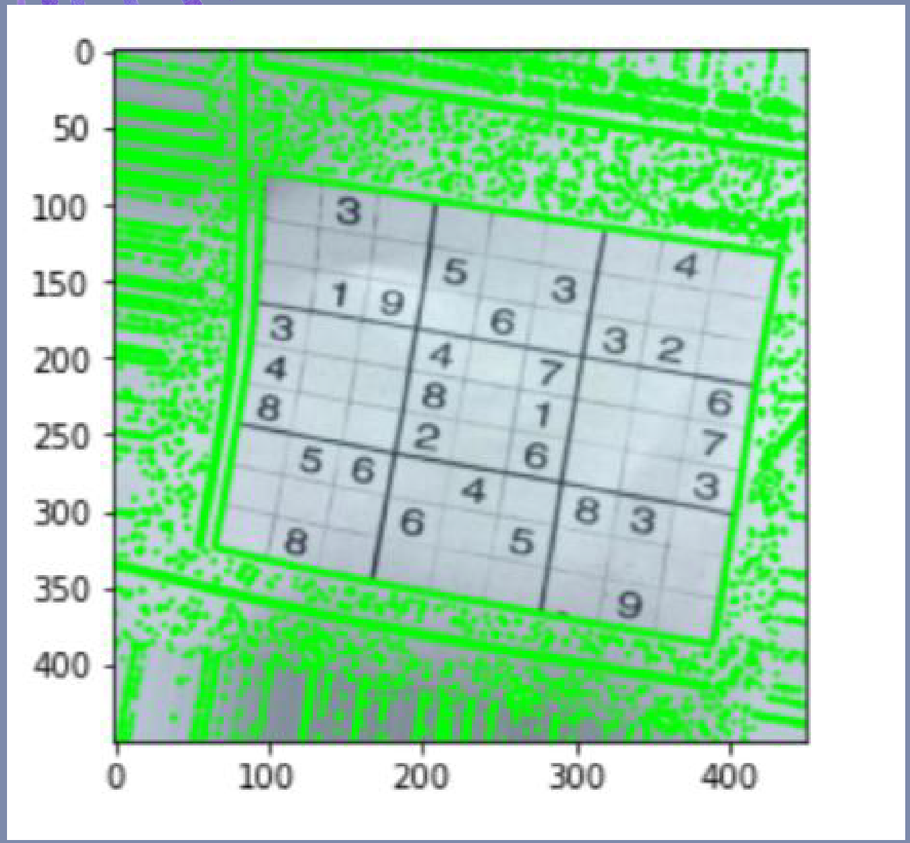
Detecting Contour

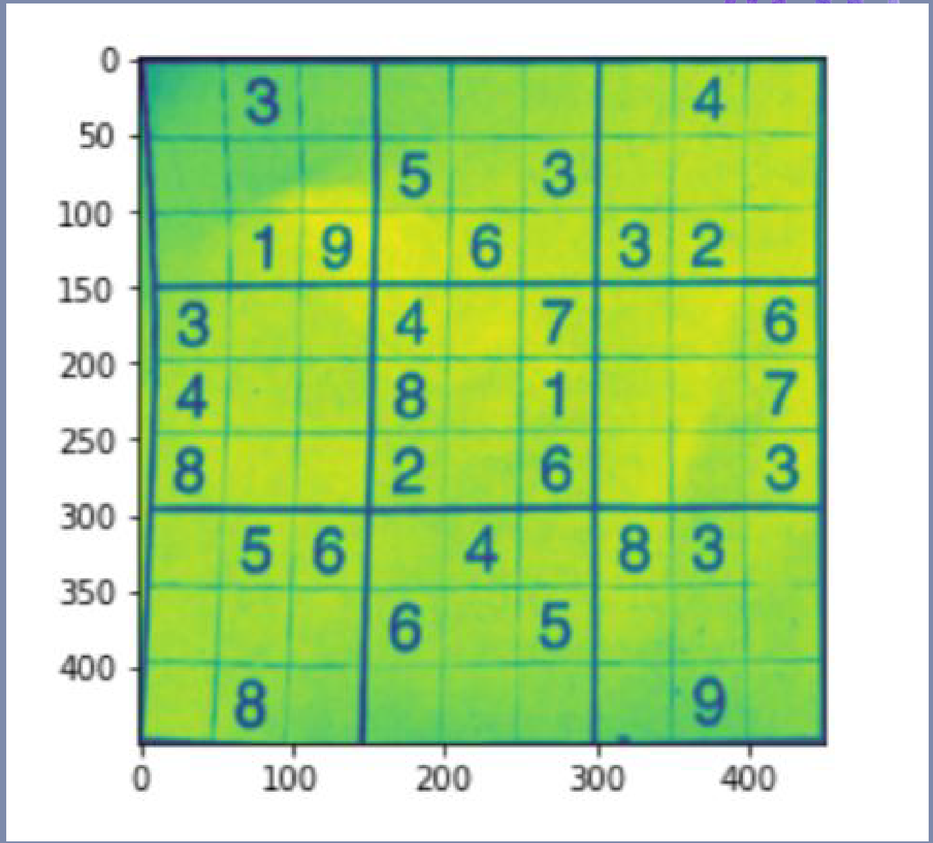
- *Now, we have a model that can perceive digits in the picture. Nonetheless, that digit recognizer doesn't do us much good in the event that we cannot find the sudoku puzzle board in a picture. We are going to detect contour. We will detect the biggest contour of the image.*

- *After applying dilation and erosion most of the noise is removed from image. We apply contour detection on image to finally segment the grid.*

- *Contours can be explained simply as a curve joining all the continuous points (along the boundary), having the same color or intensity. In our case, the contours are the individual characters. We detect all the contours and operate on contours bigger than a minimum area. We do this to neglect noisy contours that would have been detected in the image.*

After contour detection, the image looks something like this and we get our grid.



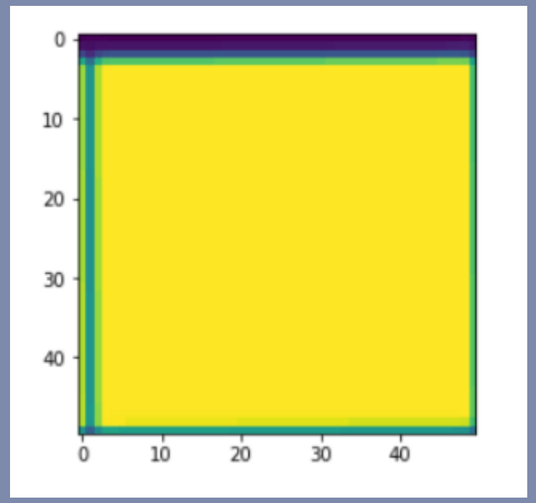


Grayscale Image

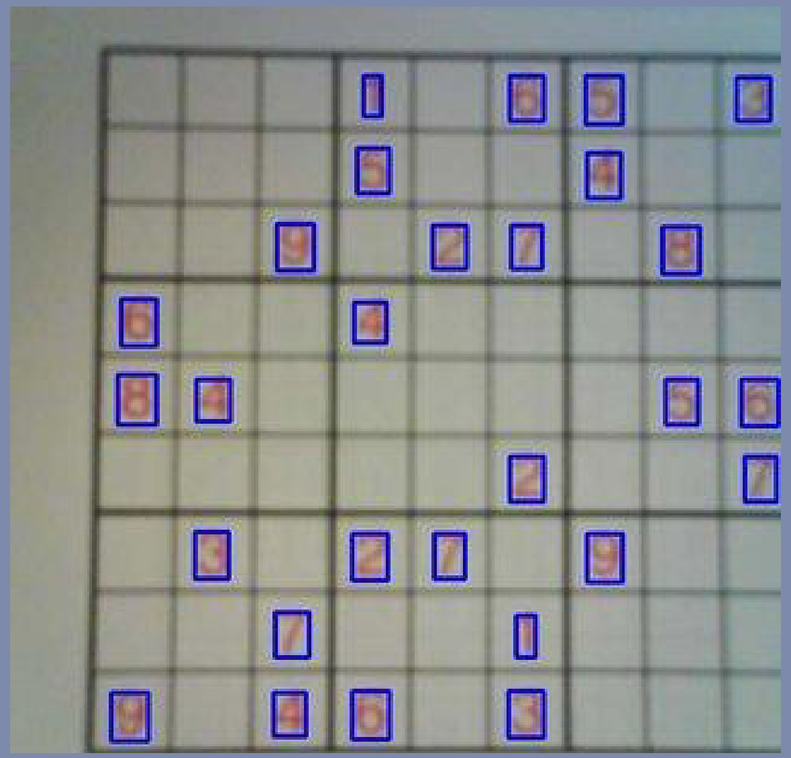
Separating Squares and cropping digits

Now, we are going to split the Square and crop the digits .

- First split the Sudoku into 81 cells with digits or empty spaces
- Cropping the cells



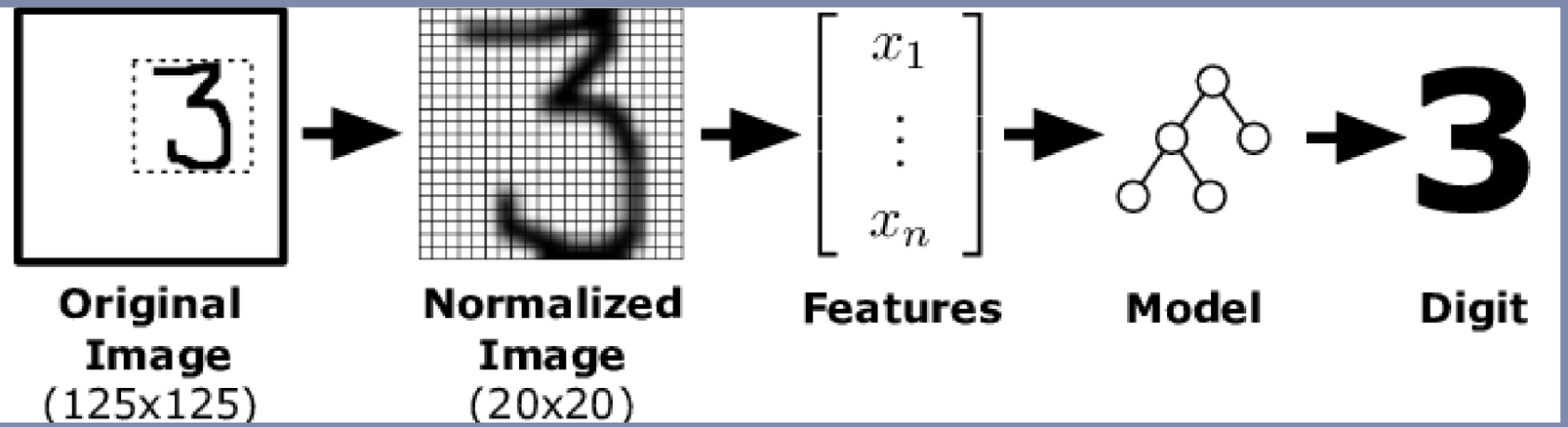
Grid

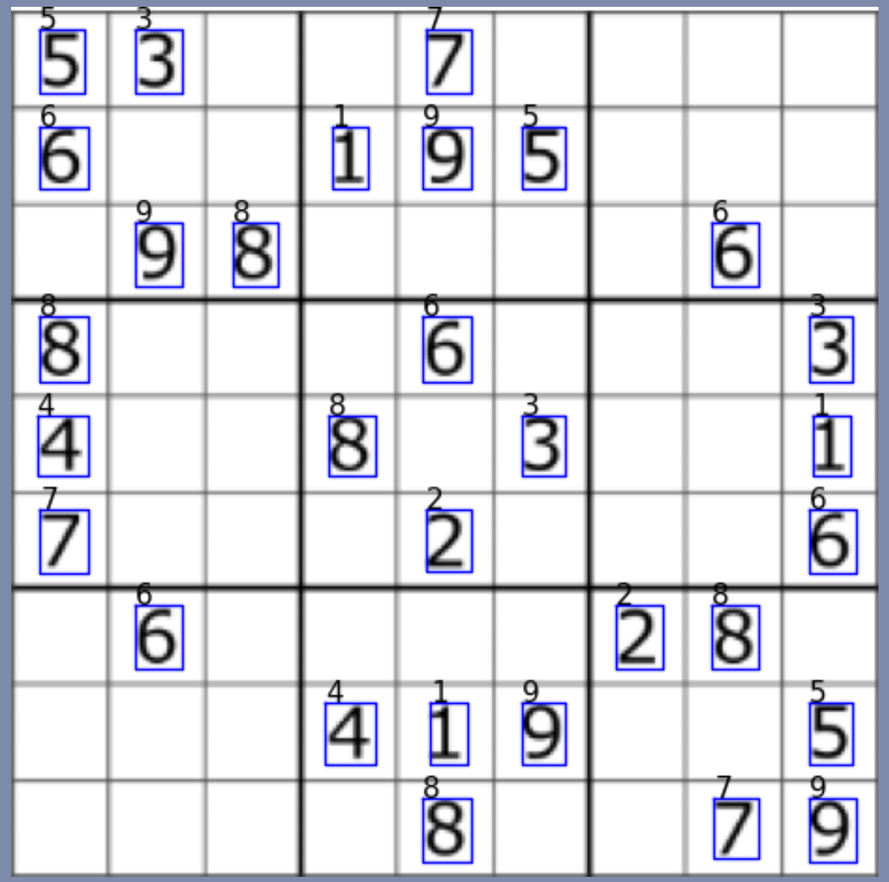


Cropping digit from grid

Digits Recognition

Now, we use our pretrained model to recognize the digits in the grid





Solving the puzzle

After recognizing the digits in the grid, we use Knuth's algorithm X to solve the sudoku puzzle and after getting the answer , we augment the result on the screen.

ALGORITHM X

1. **If there are no unsatisfied constraints remaining, the solution set is complete. Otherwise, pick an unsatisfied constraint (a column that does not contain a 1 in any of the rows in the solution set).**

2. **Pick a row that satisfies that constraint (one that has 1 at its intersection with the chosen column). If no such row exists, the solution cannot be continued. Backtrack to the previous time a row was chosen and select the next one instead.**

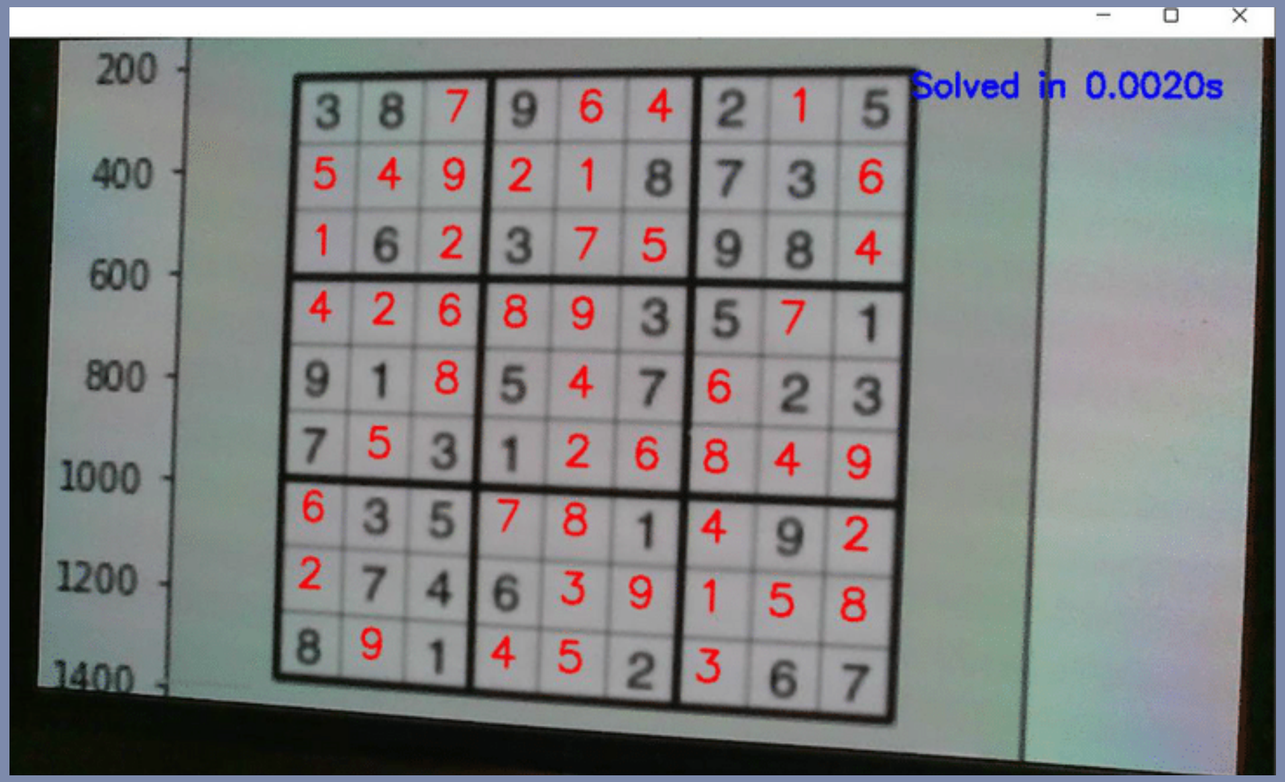
3. **Add that row to the solution set.**

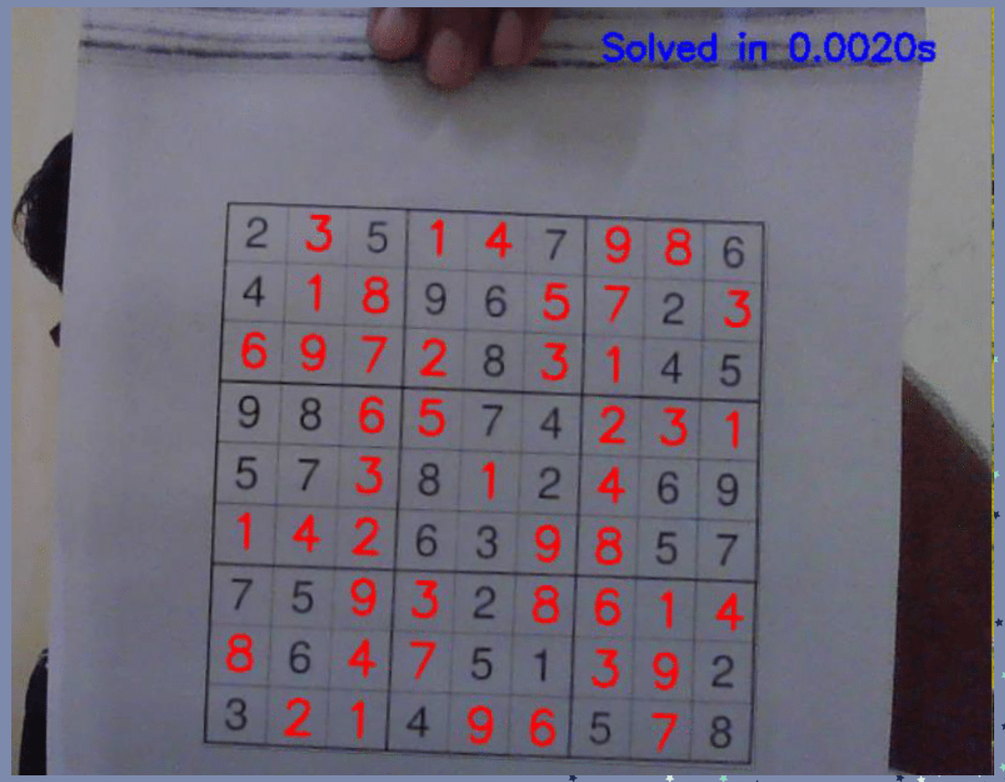
4. **Delete all rows that satisfy any of the constraints satisfied by the chosen row:that is, all rows that have 1 in the same column as any of the cells containing 1 in the chosen row.**

5. **Return to Step 1.**

Results

After Applying algo to solve the sudoku we get the final output:





Limitations

- **Our model is 85% accurate i.e., sometime our model predicts the wrong digit thus, sudoku doesn’t solve.**
- **Cannot solve puzzles that don't have a distinguishable four-point outer border.**
- **Since, our project work on real time so, lightning is important while webcam is ON.**

Conclusion

A full-fledged system for showing the solution of a Sudoku puzzle as Augmented Reality was proposed in this project. Using image processing, object localization, image classification, constraint programming and algorithm X.

THANK YOU