Handwritten Digit Recognition System

Dhruv Dogra
dd798@scarletmail.rutgers.edu

Xing Liu
xingliuhit@gmail.com

Anthony Porturas
amp357@scarletmail.rutgers.edu

***Abstract*— We will use a system that takes in image input containing a number graphic and output the value. To do so, we plan on using neural networks. The algorithmic portion of task will be a Backpropagation neural network. We plan to train and test our neural network using data sets containing numerical images.Having such a system will improve quality of life by automating recording numbers from numbers.**

I. Project Description

In this project, we intend to recognize single digit from images using neural networks. We plan to create a system that will analyze an input image and give the integer value present in it. Handwriting is unique to each person. There is variation in writing style, size of characters and even alphabet orientation. So it's a challenging task to deduct characters from images. This is a part of Deep Learning and therefore it falls in category of Massive Algorithmic.

We intend to use the MNIST dataset of handwritten digits to train and assess our system. It is a subset of larger set available from NIST(National Institute of Standard and Technology). MNIST consists of 28x28 pixel images of handwritten digits which are shown below

Fig. 1. Sample images from MNIST dataset.

It's a big database, with 60,000 training examples, and 10,000 for testing.

Challenges Ahead:

- Generating parameters
- Training the system
- Attaining high accuracy

Where our project could be used:

- Data Storage
- Education
- Historical preservation
- Textual studies

Building a single digit recognizing system is feasible within the semester. Later, this project can be expanded to work for multiple characters input as well.

The project has four stages: Gathering, Design, Infrastructure Implementation, and User Interface.

A. Stage1 - The Requirement Gathering Stage.

We plan to build a handwriting image recognition system using neural networks. In this project we plan to read an image of single numerical digit (0-9) and generate the corresponding digit in the output.

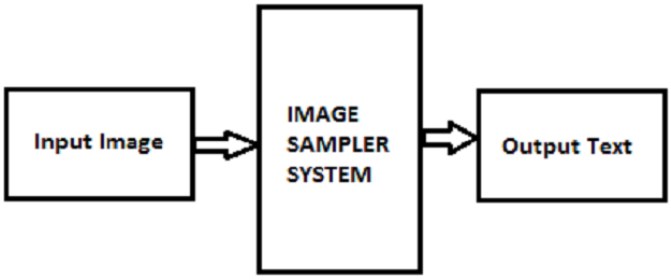


Fig. 2. Image simplifying the intended process.

To implement our system we'll be using a collection of images. This collection will be used to train and test our system to recognize the input and give the correct output.

Types of Users

We are planning this project keeping in mind the following three users:

- Policeman
- Captcha Producer
- Teacher

User Interaction Mode

We plan to create an easy to use, clean graphical interface for our project. User is expected to upload a picture containing digits to our system. He/she can use his/her mobile camera or laptop webcam to take the image of number on plain white background. On initiating 'Process' command, our system will give the desired output. We shall be providing Keyboard and Mouse functionality to our environment.

Real World Scenarios

USER: Policeman

Scenario1 description: A policeman finds that a driver drives beyond the speed limit. He takes his phone out and took a picture of the plate (we assume that there are only digits in the plate). Then our system automotive recognizes the plate and send the information of the plate to the over speeding database. Then the driver will receive a ticket.

System Data Input for Scenario1: a picture of digit plate

Input Data Types for Scenario1: picture

System Data Output for Scenario1: a recognized digit
Output Data Types for Scenario1: a recognized digit and show a picture of the value
Scenario2 description: As we all know, every road has different speed limit. in order to judge whether a driver is over speed or not, the policeman has to know the speed limit of that road. A system to recognize the speed limit from the traffic sign is necessary.
System Data Input for Scenario2: a picture of speed limit sign
Input Data Types for Scenario2: a picture of speed limit sign
System Data Output for Scenario2: the recognized speed limit
Output Data Types for Scenario2: the recognized speed limit

USER: Captcha producer
Scenario1 description: we all have the experience that we have to type the authentication code when we make the payment or logging to a website. the website will show you a digit picture. you are going to type the digits you have seen. and later, our recognition system will read the picture and compare the digits you typed with those our system recognized.
System Data Input for Scenario1: a matrix containing the pixel of the picture
Input Data Types for Scenario1: a picture containing authentication code
System Data Output for Scenario1: a number
Output Data Types for Scenario1: a picture showing the recognized value
Scenario2 description: We can collect a lot of captcha from the website. And based on our recognition system, we put the recognized number and picture into a database. Next time when you want to retrieve a picture corresponding to a specific number, then you can try to find them from the database.
System Data Input for Scenario2: collection of captcha pictures
Input Data Types for Scenario2: a number
System Data Output for Scenario2: a picture
Output Data Types for Scenario2: a picture

USER: Teacher
Scenario1 description: In math class, primary school students have to practice addition, subtraction, multiplication and division. Using our system, the teacher doesn't need to check the answer of each student. The teacher can just take the picture, our system can recognize the number and compare it with the correct answer.
System Data Input for Scenario1: a matrix containing the pixels
Input Data Types for Scenario1: a picture containing the answer
System Data Output for Scenario1: the recognized number
Output Data Types for Scenario1: the student's answer is correct or not
Scenario2 description: During each class, every student has to write his id number first. Based on system, the teacher doesn't need to match the id with the student by hand. Our system can do these things automatically.
System Data Input for Scenario2: a matrix containing the pixels
Input Data Types for Scenario2: a picture containing student id
System Data Output for Scenario2: student id
Output Data Types for Scenario2: student id

Project Timeline

TABLE I
OVERALL EXPECTED DEADLINE

Stage	*Expected Completion Date*
Stage 1: The Requirement Gathering	25^{th} October
Stage 2: The Design	8^{th} November
Stage 3: The Implementation	18^{th} November
Stage 4: User Interface	25^{th} November

TABLE II
STAGE 2: DESIGN

Phase	*Expected Completion Date*
Creating the Flow Diagram	28^{th} October
Algorithms and Data Structure Decision	1^{st} November
Writing Pseudo-code	4^{th} November
Analysis of Constraints	6^{th} November
Review and Evaluation	$8^{th} November$

TABLE III
STAGE 3: IMPLEMENTATION

Phase	*Expected Completion Date*
Sample Small Data Snippet	9^{th} November
Sample Small Output	13^{th} November
Working Code	14^{th} November
Demo and Sample Findings	16^{th} November
Review and Evaluation	$18^{th} November$

TABLE IV
STAGE 4: USER INTERFACE

Phase	*Expected Completion Date*
Making User-Friendly Interface	23^{rd} November
Review and Evaluation	$25^{th} November$

Division of Labor

TABLE V
STAGE 2: DESIGN

Phase	*Responsibility*
Creating the Flow Diagram	Anthony
Algorithms and Data Structure Decision	Dhruv
Writing Pseudo-code	Xing
Analysis of Constraints	Anthony
Review and Evaluation	Dhruv

TABLE VI
STAGE 3: IMPLEMENTATION

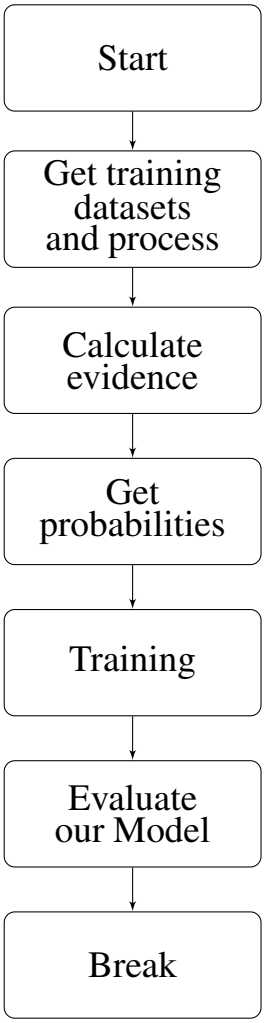
Phase	*Responsibility*
Sample Small Data Snippet	Xing
Sample Small Output	Dhruv
Working Code	Anthony
Demo and Sample Findings	Xing
Review and Evaluation	Dhruv

TABLE VII
STAGE 4: USER INTERFACE

Phase	*Responsibility*
Making User-Friendly Interface	Anthony
Review and Evaluation	Dhruv

B. Stage2 - The Design Stage.

- **Flow Diagram**



- **Short Textual Project Description**
  Each block shown above has been explained below.
  - *Block 1: Get an input image and process it*
    MNIST datapoints is made of 2 parts: image and corresponding label. Image in the dataset is present as a 2D array of size 28*28. The image can be visualized as shown below:

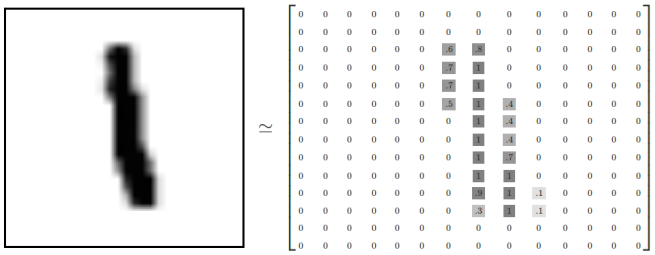


Fig. 3. Image and its corresponding 2D Array

The numbers in the array define pixel intensity at that point. These arrays are flattened into a 784-dimensional vector. The label associated with each image is also processed and converted into a vector of desired form. The labels store the integer which describe the corresponding image. We shall store the vector as "one-hot" vector, which means that it's value is 1 in one dimension and is 0 in every other dimension. For example, 2 would be represented as [0,0,1,0,0,0,0,0,0,0].

  - *Block 2: Calculate evidence*
    This step involves creating "evidence", which is used to decide to which group our image belongs. To calculate evidence we perform weighted sum of the pixel intensities. To this a "bias" is added as well. What bias does is that some output are more likely than others. Mathematically we can express evidence as follows: Evidence[i] = Summation(Weights*Input + Biases)
  - *Block 3: Get probabilities*
    We convert the previously calculated evidence is used to get probabilities. The evidence is fed to "activation" function which generates a probability distribution over 10 digits. The softmax function is defined as follows-

$$softmax(x)_i = \frac{exp(x_i)}{\sum_j exp(x_j)}$$

Softmax regression can be shown as follows-

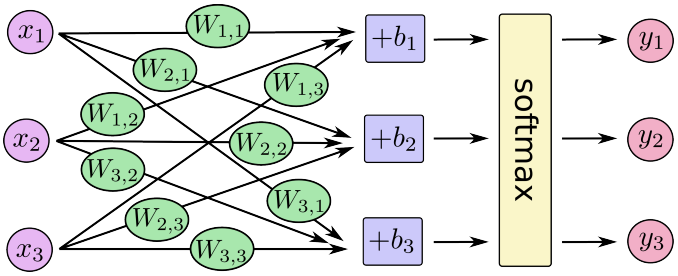


Fig. 4. Softmax Graphical Representation

In terms of equation we can represent it as-

$$\begin{bmatrix} y_1 \\ y_2 \\ y_3 \end{bmatrix} = \text{softmax}\left(\begin{matrix} W_{1,1}x_1 + W_{1,2}x_2 + W_{1,3}x_3 + b_1 \\ W_{2,1}x_1 + W_{2,2}x_2 + W_{2,3}x_3 + b_2 \\ W_{3,1}x_1 + W_{3,2}x_2 + W_{3,3}x_3 + b_3 \end{matrix} \right)$$

Fig. 5. Softmax Equation

- *Block 4: Training* In this step our motive is minimize our cost function. First, we cost function as a "cross entropy", which is defined as follows-

$$H_{y'}(y) = -\sum_i y'_i log(y_i)$$

Here y is our predicted probability distribution, and y' is the true distribution. Backpropagation and gradient descent algorithm is applied in order to minimize the cost.

- *Block 5: Evaluate our Model* In this step we compare our group with highest predicted probability against the label.

- **High Level Pseudo Code System Description.**
Psuedo code is as follows-

```
int [] inspectCaptcha(int[][] image)
    if (image.length == 0 —— image[0].length == 0)
        Return null;
    int [][][] charactersArray = findCharacters(image);
    int[] evidence = charactersArray[0][0].length;
    for (int i = 0; i ¡ evidence.length; i++)
        Evidence[i] = summation(weights*input + bias);
    int [] probability = new int[character set size];
    int[] output = new int[evidence.length];
    for (int j = 0; j ¡ evidence.length; j++)
        Probability = softmax( charactersArray, evidence[j] );
        Char character = characterList( indexOfMax( probability ) );
        Output[j] = character;
    Return output;
```

- **Algorithms and Data Structures.**
  - *Gradient Descent*
    This is an optimization algorithm which tells us whether to increase or decrease the weights in order to get a lower cost on the next iteration. Derivatives are used to implement this. It gives us the slope of the function at a given point, and from the slope we get the direction (sign) to move the coefficient values.
    The procedure starts off with initial values for the coefficient or coefficients for the function. These could be 0.0 or a small random value.
    coefficient = 0.0
    The cost of the coefficients is evaluated by plugging them into the function and calculating the cost.
    cost = f(coefficient)
    The derivative of the cost is calculated.
    delta = derivative(cost)
    A parameter (alpha) is introduced to control the degree by which coefficients can change after each update. Updated coefficient value is
    coefficient = coefficient - (alpha * delta)
    This process is repeated until the cost of the coefficients (cost) is as small as possible.

  - *Backpropagation algorithm*
    It is use to get the derivative of our neural network's cost function w.r.t. its weight. The algorithm can divided into 2 main parts as follows-
    Feed-forward: the input x is fed into the network. The primitive functions at the nodes and their derivatives are evaluated at each node. The derivatives are stored.
    Backpropagation: the bias is fed into the output unit and the network is run backwards. Incoming information to a node is added and the result is multiplied by the value stored in the left part of the unit. The result is transmitted to the left of the unit. The result collected at the input unit is the derivative of the network function with respect to x.

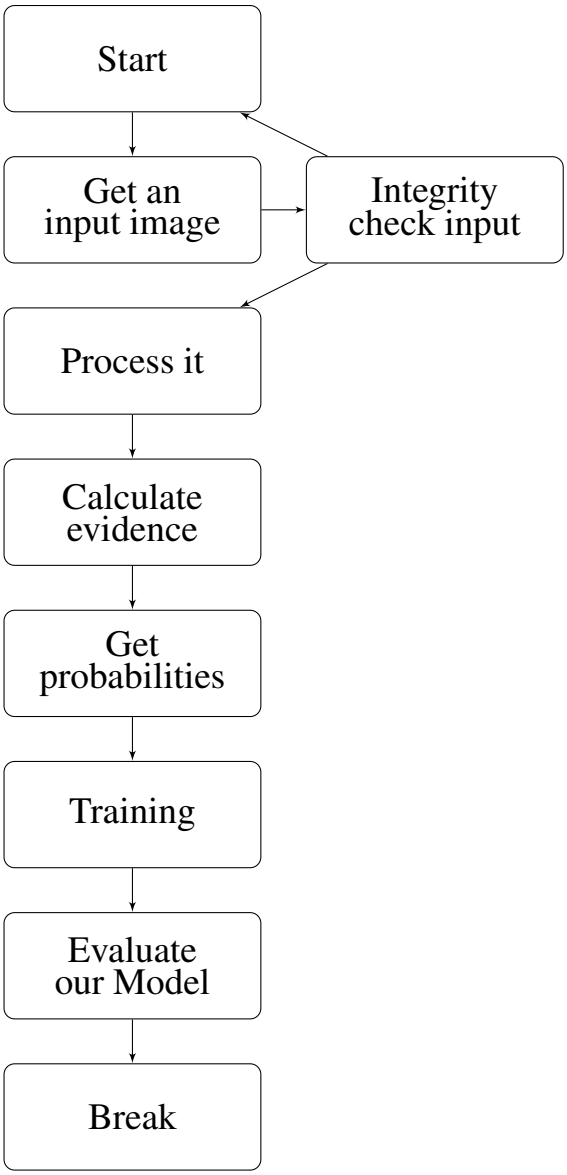
  - *2D and 3D Arrays as the data structures*
    These are used to store and carry out operations on input images.

- **Flow Diagram Major Constraints.**
Integrity constraints are listed below:

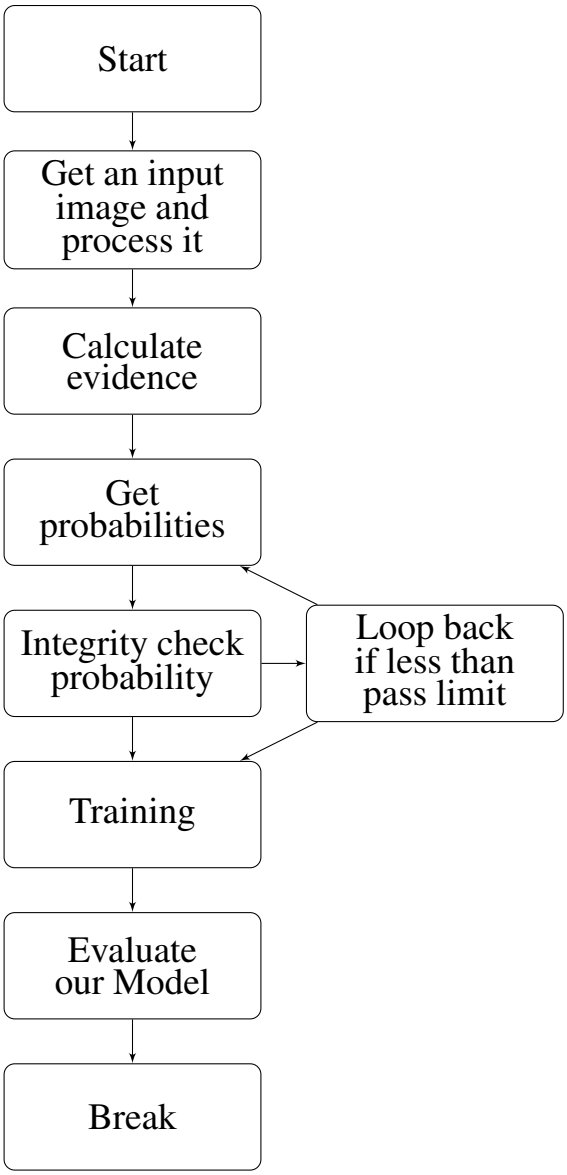
  - Integrity Constraint: input The input is a 2D array of integers from 0 to 255. This is important to make sure that the program is running with correct inputs, otherwise it would not correctly work. The integer values represent the pixel value, so the possible range is 0 to 255. The array has to be 2D for signify length and width.

  - Integrity Constraint: probability threshold A constraint is put on selecting the highest probability. The highest probability has to be above a certain threshold to be considered accurately found. This is to increase correctness and not let bad values pass. If it doesn't pass the threshold, the calculations are repeated until the pass limit is reached. Then it will ignore the character.

  - Flow Diagram with input constraint



– Flow Diagram with probability constraint



C. Stage3 - The Implementation Stage.

Language used to train and test neural network is Python. To get input and display output we used Javascript, HTML and CSS. Sublime text editor along with Anaconda IDE was used write and compile our source code. To visualize the input matrix we used Python.

- Sample small data snippet.
  From MNIST dataset we get three lists, lets call them *training list*, *validation list* and *test list*. All three list contain 2-tuples (x,y) where x is an 784-dimension array containing input image and y is a 10-dimension unit vector corresponding to the correct digit for x. Training list has 50,000 tuples and other 2 have 10,000 tuples.

  The format in which we received one input image in 784-dimension and is written in inputMatrix.txt file. It is a matrix of size 28*28 with elements values ranging between 0 and 1.

Format of the unit vector is-
0 0 0 0 0 1 0 0 0 0
On visualizing the image in mentioned text file we get the following output-

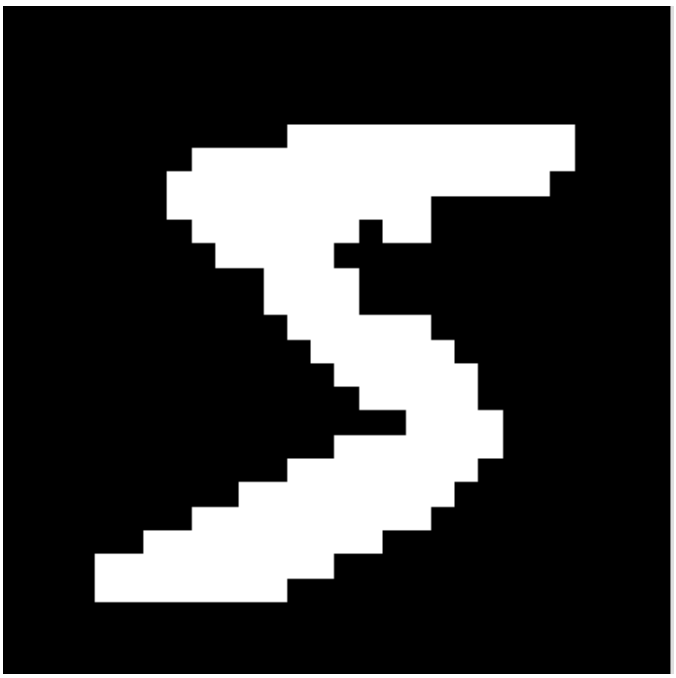


Fig. 6. Visualizing of the input matrix in Matlab

The value represent by them is 5.

- Sample small output
  The neural network processes the image entered by the user and assigns percentage against ten numbers ranging from 0-9. The percentage indicates which number is most likely drawn by the user. We output the number with the highest percentage among all the 10 numbers. The output image is shown below-

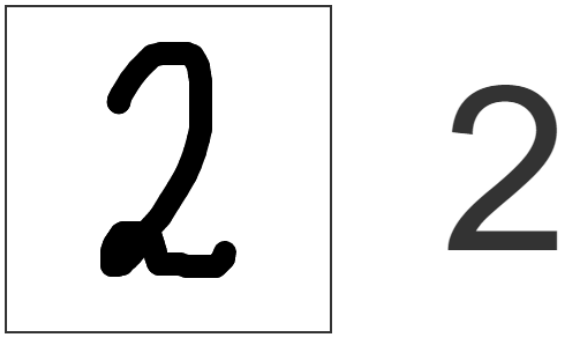


Fig. 7. Output from the Neural Network

The accuracy obtained after training and testing neural network is as follows-
For number 0 accuracy is : count 980 correct 954 0.973469387755
For number 1 accuracy is : count 1135 correct 1114 0.981497797357
For number 2 accuracy is : count 1032 correct 936 0.906976744186
For number 3 accuracy is : count 1010 correct 913 0.90396039604
For number 4 accuracy is : count 982 correct 936 0.953156822811
For number 5 accuracy is : count 892 correct 816 0.914798206278
For number 6 accuracy is : count 958 correct 895 0.934237995825
For number 7 accuracy is : count 1028 correct 962 0.93579766537
For number 8 accuracy is : count 974 correct 877 0.900410677618
For number 9 accuracy is : count 1009 correct 917 0.90882061447
The average accuracy comes out to be 0.931312 or 93%.

- Working code
  The code can be divided into two main parts-
  1) Training and testing neural network
     Code required to train network is written in Python. It gives weights and biases which for all the 25 "neurons" present in the hidden layer.
  2) Displaying output
     Web development languages are used to get output. Parameters obtained from above code are inserted in Javascript file and then neural network is run.
- Demo and sample findings
  - *Data size:* Our algorithm takes content from compressed dataset and uses it to train and test the Neural Network. The size of compressed file is 16.2 Megabytes(Mbs). At the start of the algorithm we load the entire dataset at once in the memory.
  - MNIST dataset is special as it is a advanced version of NIST dataset and widely used to train image processing systems. In this project we managed to get an accuracy of atleast 93% using a single hidden layer having just 25 neurons.

D. Stage4 - User Interface.

The user interface is minimal, keyboard accessible and user-friendly. The user has two options where he can either-

1) Upload an image
2) Draw an image

On pressing the Recognize button we just display the answer received from the Neural Network. The short deliverables for this stage have been addressed below :

- !!!!The modes of user interaction with the data (text queries, mouse hovering, and/or mouse clicks ?).
- The user can interact with the data by clicking on an upload button or drawing in the canvas by holding

down the left mouse button. The buttons work by clicks. ENTER and Spacebar click also triggers the click.

- !!!!The error messages that will pop-up when users access and/or updates are denied
- Two errors can take place and have been explained in later section.
- !!!!The information messages or results that will pop-up in response to user interface events.
- When the user uploads an image, the image will be uploaded onto a canvas to the left. After running the program, the recognized digit will appear on the canvas to the right. A pop appears which can is used to record if correct answer is given by the network or not. After clearing the canvases, the canvases will turn blank.
- !!!!The error messages in response to data range constraints violations.
- The user will get an error when attempting to upload a file that isn't an image file type.
- !!!!The interface mechanisms that activate different views in order to facilitate data accesses, according to users' needs.
- There is one single interface that holds everything necessary.
- !!!!Each view created must be justified. Any triggers built upon those views should be explained and justified as well. At least one project view should be created with a justification for its use.
- We implemented one view that holds the project and its functionality. We launch dialog boxes in case errors happen and also at the end to record the response. We aimed for simplicity and so, our view is minimalist and visually appealing.

Deliverables for Stage4 as follows:

- The initial statement to activate your application with the corresponding initial UI screenshot
- The application is activated by running project.html with all of the files in the same folder. It will bring up the main and only screen needed.

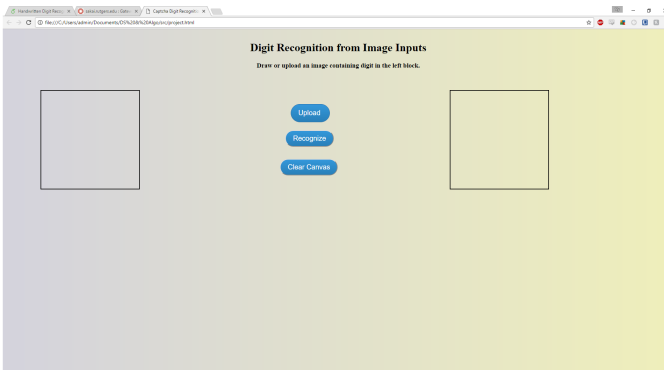
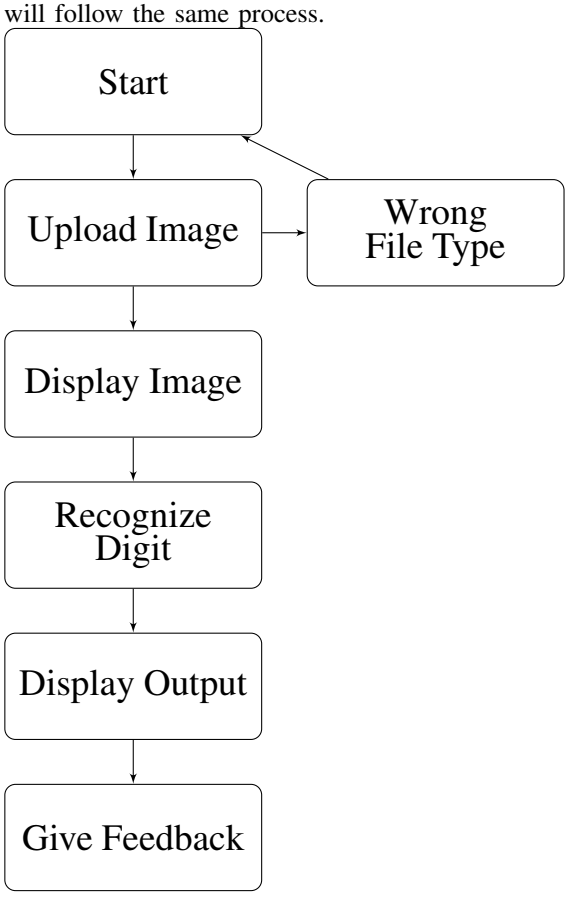
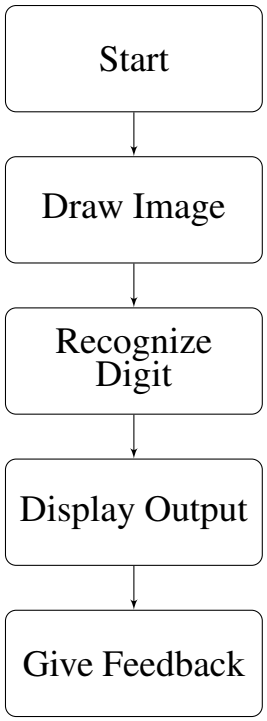


Fig. 8. The initial screen

- Two different sample navigation user paths through the data exemplifying the different modes of interaction and the corresponding screenshots.
- The user may upload an image containing a single digit and press recognize. The program will display the image on the left and the corresponding recognized image on the right. The user may also draw on the canvas to the left in place of uploading an image file. Hitting recognize will follow the same process.





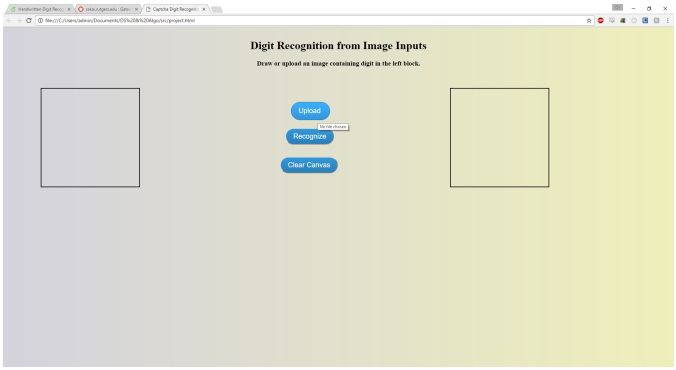


Fig. 9. Upload: Step 1. Click Upload

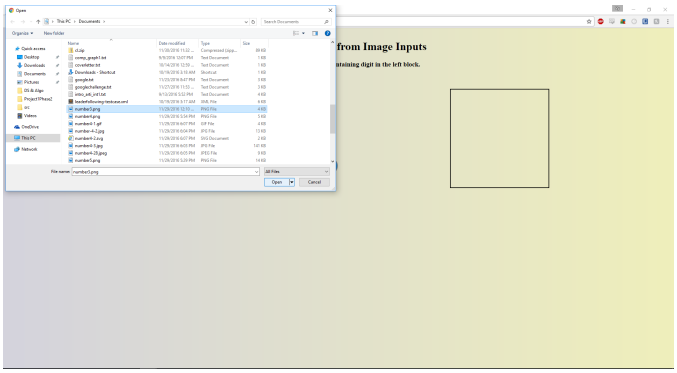


Fig. 10. Upload: Step 2. Select file

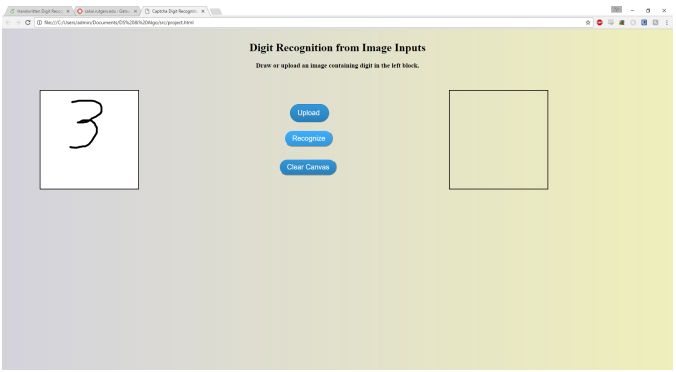


Fig. 11. Upload: Step 3. The image is uploaded to the left canvas

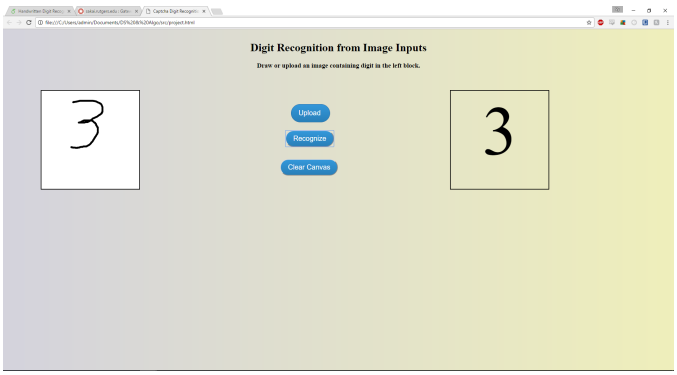


Fig. 12. Upload: Step 4. Click recognize, look at results

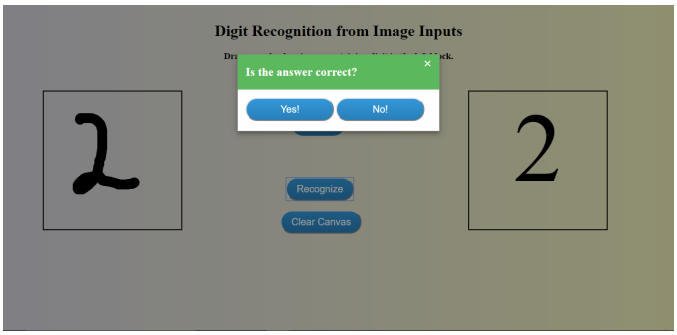


Fig. 13. Draw: Step 5. Provide Feedback

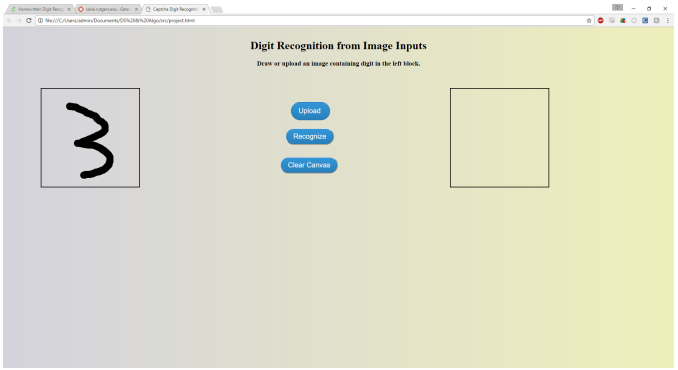


Fig. 14. Draw: Step 1. Draw an image on the left canvas using mouse

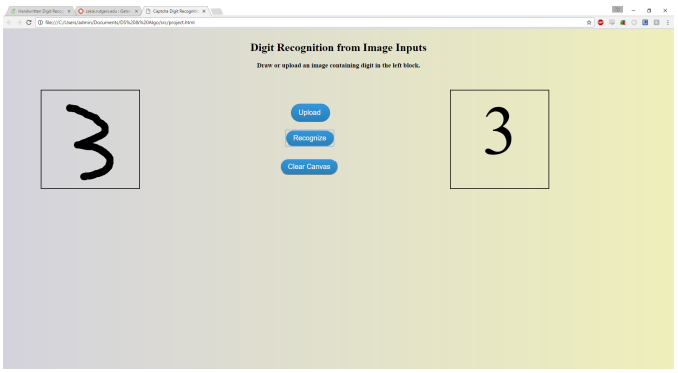


Fig. 15. Draw: Step 2. Click recognize, look at results

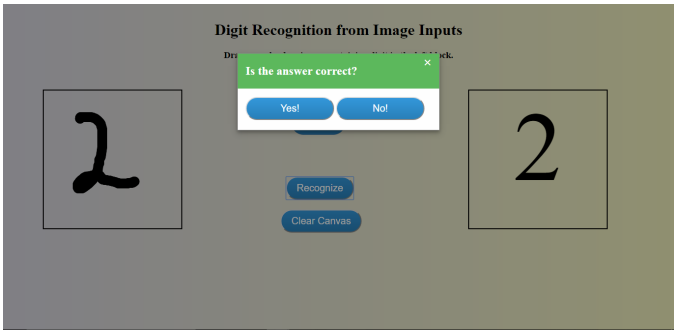


Fig. 16. Draw: Step 3. Provide Feedback

- Two main errors that can occur while user is using the system have been dealt below:

  1) Error 1

     – *Error Message:* Please enter a valid image.
     – *Explanation:* This error occurs when the user enters an invalid file like say PDF or text file.
     – *Scenario:* Say a teacher is using it and he/she may inadvertently select an incorrect file, instead of image, in File dialog box.

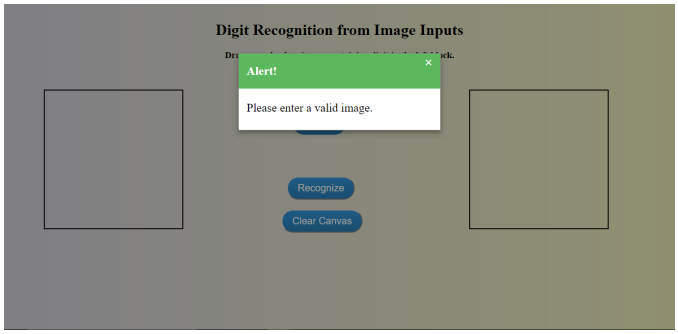


Fig. 17. Error Message 1: Please enter a valid image.

  2) Error 2

     – *Error Message:* Please draw or upload an image.
     – *Explanation:* This error happens when the left block is empty and user presses the Recognize button.
     – *Scenario:* A student while testing the system can press Recognize which may give some erroneous answer. This alert pop ensures that inexplicable output are given.

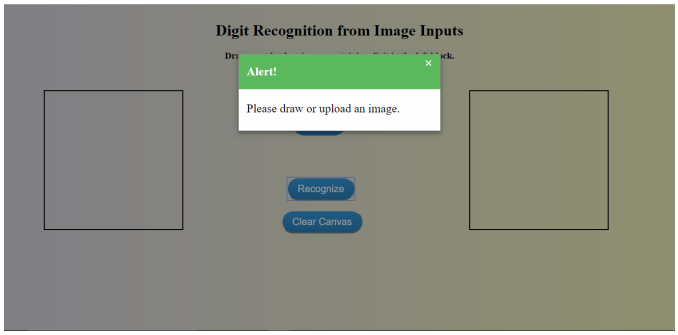


Fig. 18. Error Message 2: Please draw or upload an image.

These error message can be closed by clicking anywhere on window or by clicking the **Close** symbol present in the top right corner.

- After user correctly uploads or draws an image and presses Recognize, we get the output of Neural Network in right hand side box. An alert is also displayed at the end so as to record the performance of our system.

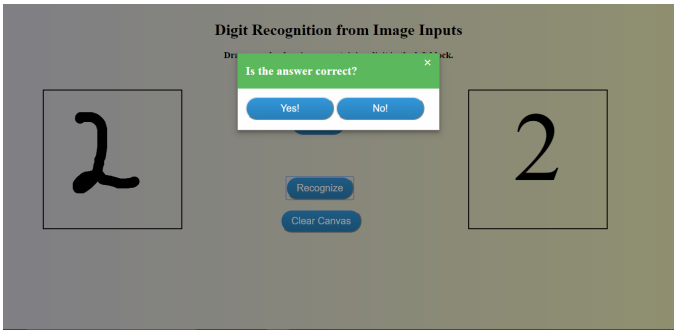


Fig. 19. Message Box obtained at the final step

  - *Information message:* Yes/No dialog box.
  - *Explanation:* Purpose of this message box is improve the incorrect answers given by the Neural Network. We expect user to select *NO* when wrong answer is given. In this case we shall save the answer by network and the image input in a JSON. Additionally this JSON object is outputted in console. We intend to analyze these inputs and use to them to improve our accuracy
  - *Scenario:* User may draw random image or upload an image whose pixel configurations we have not handled in the system.

- There are two ways by which user can perform on this system, namely Keyboard and Mouse.
  - *Using Mouse:* Using mouse, user can select all the buttons necessary to perform required actions.
  - *Using Keyboard:* Keyboard pathways also work in our system. By pressing TAB or SHIFT+TAB, user can move in the system. Additionally we can press F6, to get focus on the buttons. A highlight appears when focus is on buttons letting user know where the focus is present.

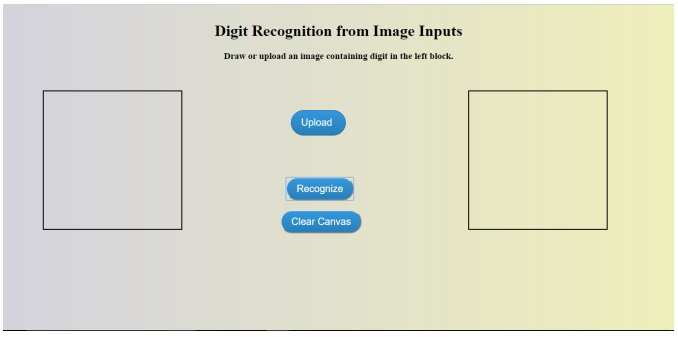


Fig. 20. Recognize button with focus.

To select a button Spacebar and ENTER key can be pressed.