MNIST Digit Recognition on DE1-SoC for Final Project

Created by: Irwin Ngo & Liam Ngo
Teaching Assistant: Isidor Kaplan
Date: April 13th, 2023
Course: ECE243
Section: 0103

1.0 Introduction

The software recognizes digits (0 to 9) drawn by the user on a 28x28 pixelated canvas with a mouse. To do so, we programmed a linear neural network model in C-language that takes in 784 inputs (28x28) and outputs 0 to 9. The program is run on the DE1-SoC board, and the predicted result is displayed on a HEX segment.

This document lists the following:

- Instructions on deploying and running the software
- Attribution report on the software's creation

2.0 Software Instructions

The following instructions go through four main steps, which involves:

- How to execute the program
- How to start the program
- How to navigate the program

2.1 Executing the Program

The file to execute the software is called "combined.c". To run it…

1) Create a new project in the Quartus Monitor Program
2) Select the ARM Cortex-19 Architecture, then click Next
3) Select the "DE1-SoC Computer" as the system, then click Next
4) Select the "C Program" as the program type, then click Next
5) Add the "combined.c" to the source files
6) Add "-std=gnu99" to the "Additional compiler flags"
7) Add "-lm" to the "Additional linker flags", then click Next
8) Ensure that the DE1-SoC is on and connected. Select it as the Host Connection, and click Next.
9) Click save, then compile and load the code.

After loading, you should be able to run the program with the toolbar above.

2.2 Using the Program

The user must first connect their PS2 mouse to the DE1-SoC board (after it runs). They will only be using the mouse to interact with the program. They will interact in four pages:

1) Start Page
2) Load Page
3) Menu Page
4) Canvas Page

The user will proceed through these pages in an ascending order, starting from the start page.

2.2.1 Start Page

The start page, illustrated in Figure 1, has two buttons:

- Train Model
- Load Model

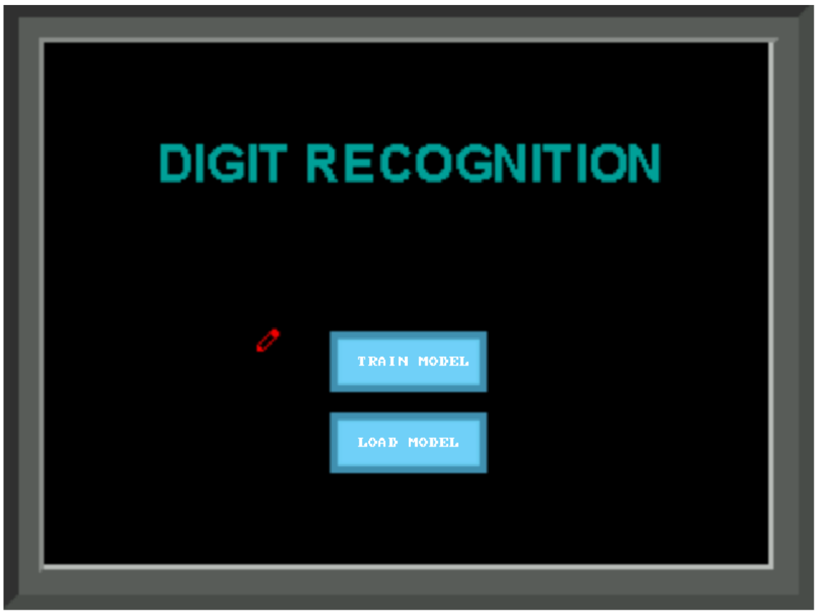


Figure 1. Image of the Start Page

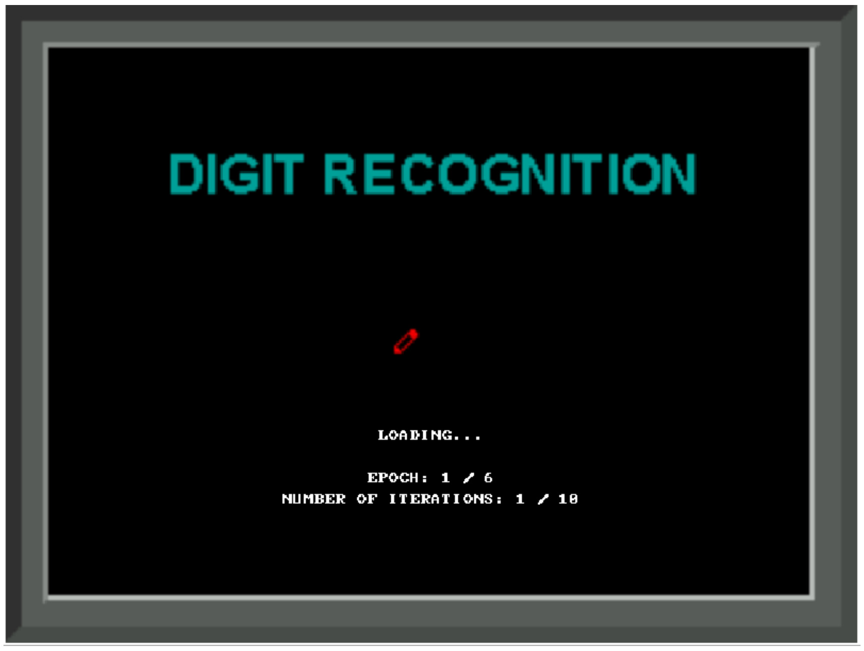
The “Train Model” button trains the neural network model with a dataset of 3000 examples from the MNIST database. Due to the processor speed, it takes approximately 40 minutes to train the model.

The “Load Model” button loads the pre-trained model that is trained with a dataset of 60000 examples from the MNIST database. Unlike training, loading the pre-trained model takes a few seconds.

Both of the buttons lead to the “Load page”, which is when the model is trained or getting loaded.

2.2.2 Load Page

The load page, illustrated in Figure 2, indicates that the model is being trained or loaded. In this page, the user cannot move their cursor. If the model is being trained, the loading screen will display the number of epochs and the number of iterations remaining. Otherwise, it will simply display “Loading…”



LOADING...

EPOCH: 1 / 6
NUMBER OF ITERATIONS: 1 / 10

Figure 2. Image of the Load Page with training

After the model finishes being trained or loaded, the user will automatically be brought to the Menu Page.

2.2.3 Menu Page

The menu page, illustrated in Figure 3, has two buttons;

- Draw button
- Exit button

The "Draw" button brings the user to the canvas, where the user can draw a digit and let the program predict what digit was drawn in a 28x28 pixelated canvas.

The "Exit" button stops the program from running with "exit(1)" in the C program.

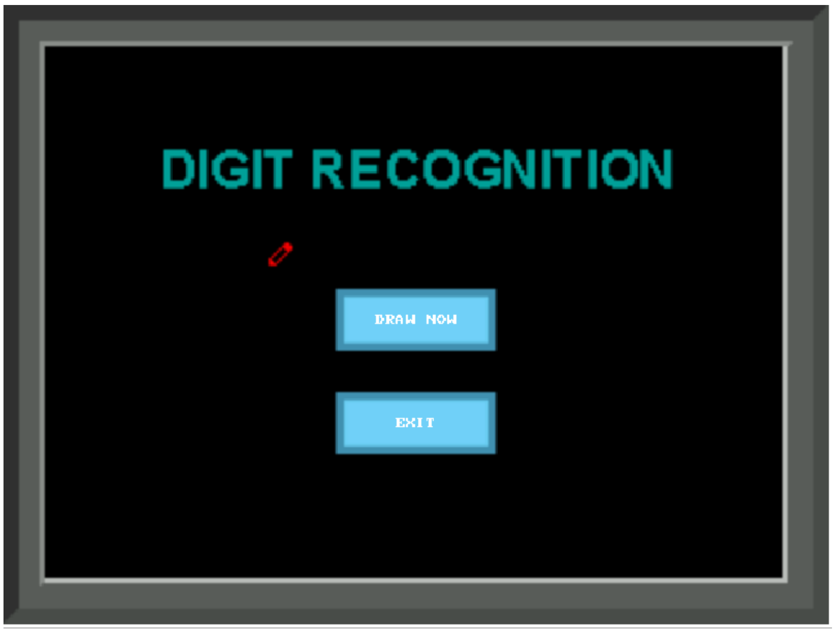


Figure 3. Image of the Menu Page

2.2.4 Canvas Page

The canvas page, illustrated in Figure 4, has three buttons:

- Back button
- Draw mode button
- Recognize button

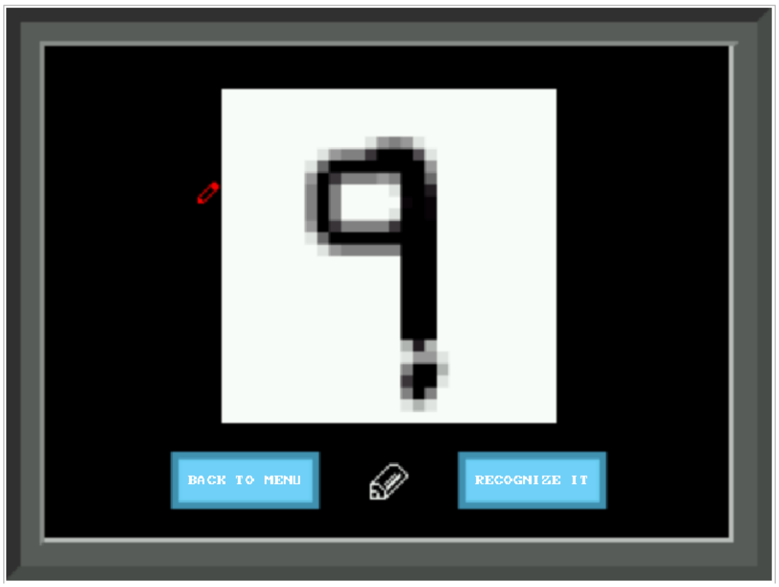


Figure 4. Image of the Canvas page.

The “Back” button clears the current canvas and returns the user to the menu page. It resets the entire drawing, such that when the user goes back to the “Canvas” page, it shows a white canvas.

The “Draw mode” button is in the middle, depicted by a “Pencil” or “Eraser” icon, as shown in Figure 5. If the current icon is a “Pencil”, the user can draw in black. If the current icon is an “Eraser”, the user can erase their drawings.

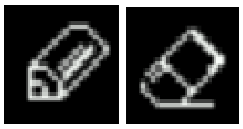


Figure 5. Icons of the Pencil (left) and Eraser (right) drawing mode

To toggle between the two modes, the user can click on the displayed icon. To draw or erase, the user must hover their cursor over the canvas, hold left click, and move their mouse around.

The “Recognize” button starts the prediction of the digit. After predicting, the number is displayed on the 0th HEX segment on the DE1-SoC board.

3.0 Attribution Report and Software Architecture

The software involves 2 major components:

- Neural Network model for Digit Recognition
- Graphics Rendering & User Input

3.1 Neural Network Model

Irwin has done the following:

- Implemented Linear layer and node functions
- Implemented Random weight/bias initialization function
- Implemented ReLu, Leaky ReLu, and Softmax activation layer functions
- Implemented Cross Entropy Loss functions
- Implemented gradient calculation functions for all layers
- Implemented the forward- and backward-propagation for training
- Improved gradient descent by implementing stochastic gradient descent
- Ensured no memory leak with the many memory allocations and deallocations
- Tweaked hyper-parameters to maximize accuracy and minimize training time
- Created a struct to hold model data
- Implemented functions that allows programmer to freely build the model's input size, layers, activation functions, and loss functions
- Tested various model layers and number of nodes
- Implemented function to save model weights & bias to a header file
- Implemented function to load model weights & bias from a header file
- Created separate code that writes a specific number of train and test examples to a header file

As a result, the current neural network model is as follows:

1) Input: Double array of size 784
2) Linear Layer: 98 nodes with Leaky ReLu activation layer
3) Linear Layer: 10 nodes with Softmax activation layer
4) Output: Single array of size 10

3.2 Graphics Rendering and User Input

Irwin has done the following:

- Implemented all four pages (Start, Load, Menu, Canvas)
- Photoshopped the button.png and buttonHover.png
- Drew the "Pencil" and "Eraser" icons as .png files
- Created python code that converts .png files to 2D RGB arrays in the format of the DE1-SoC VGA using the OpenCV library
- Drew the "DIGITS RECOGNITION" title as a .png file
- Drew the "cursor" icon as a .png file
- Implemented and/or completed handle events for all the button hovers and clicks
- Optimized rendering of graphic components (cursor, canvas drawing, buttons) by selectively re-rendering pixels at specific orders
- Wrote every rendering function with consideration of double-buffering

- Implemented character buffering and wrote all the texts
- Implemented the interrupts for the PS2 mouse
- Completed the PS2 byte reading for mouse and implemented the mouse movement

Liam has done the following:

- Implemented mouse control with PS/2 port and draw 3x3 pixels on the 28x28 canvas
- Implemented Load Model button
- Implemented the pixels’ colours when drawing on canvas