ECE243 Project Report - Handwritten Digit Classification

Yun-Ta (Daniel), Lee / Lily Chen

Project Description:
This project implements handwritten digit classification on the DE1-SoC FPGA using the NIOS V soft processor. Users can draw digits (0-9) on a VGA display using controls from the PS/2 keyboard. The drawn image of 240x240 pixels down sampled using bilinear resizing to a 28x28 pixel grayscale image, which is then passed through a modified LeNet-5 CNN model for classification. Side panel of the VGA display (80x240 pixels) is used to show the probability of each digit. The predicted digit (highest probability) is displayed on the 7-segment HEX display. Different keys are used for controlling the program's execution.

Keyboard controls:
Our program polls for keypress packets from the PS/2 port then executes different functions according to the corresponding values received as shown below:

- Space bar: start program
- Arrow keys: move and draw on canvas
- WASD keys: move cursor on canvas
- Backspace: erase current cursor position
- ESC: clear / reset canvas
- Enter: submit current image for classification
- Delete: back to start screen

Block Diagram:

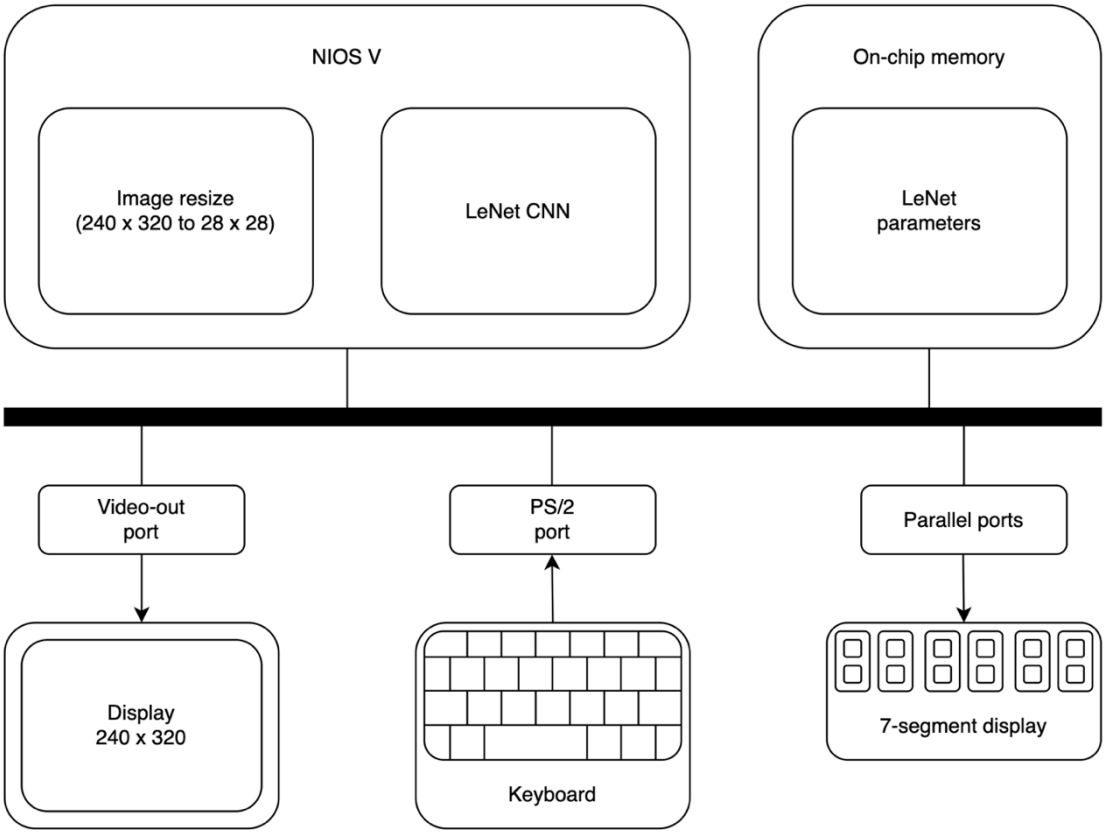

Fig. 1: block diagram of project

Project Operation:
Users are first greeted with a start screen that displays the name of the project with an image of the program interface as shown in Fig. 2. It also shows the controls for the PS/2 keyboard that is used for movement control and model inference.

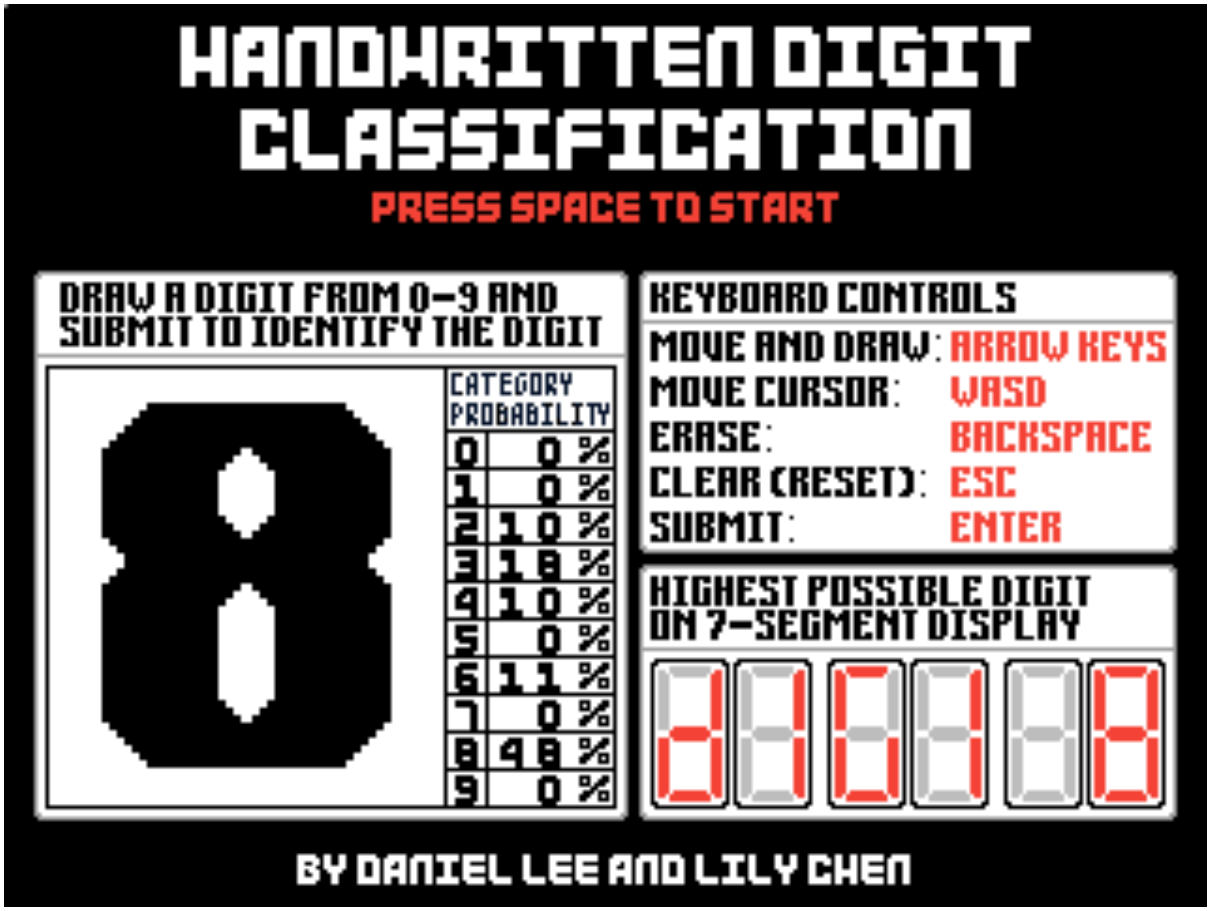

Fig. 2: start screen

After pressing the space bar, the main program starts and the display switches to Fig. 3. Users can use the arrow keys to draw on the canvas or use the WASD keys to change the cursor position without drawing. The backspace key is used for erasing and the esc key is used to reset the entire screen. The pen size is a pixel circle with radius of 16 pixels while the current cursor position is shown by a 1-pixel red dot on the screen. When users are finished with their handwritten digits, the enter key is used to submit the image on the canvas to the CNN model for inference. The program captures the pixel values and passes it through bilinear resizing from 240x240 pixels to 28x28 pixels. The 28x28 grayscale pixels are passed into the model as input and the 10 output nodes represents the value of each digit (from 0-9) being the correct one, with positive representing a higher probability and vice versa for negatives. The program then displays the probability of each digit by treating all negative numbers as 0's and simply calculating the percentage out of the total sum. This calculation method is not an ideal representation of the real probability of each number, but it gives users a more visual representation without extensive calculations (like using SoftMax). The digit with the highest value is displayed on the 7-segment hex display for a clear classification. The program continues to run so users can choose to either further modify the image on the canvas or clear the screen and restart with the esc key as we poll for key presses.

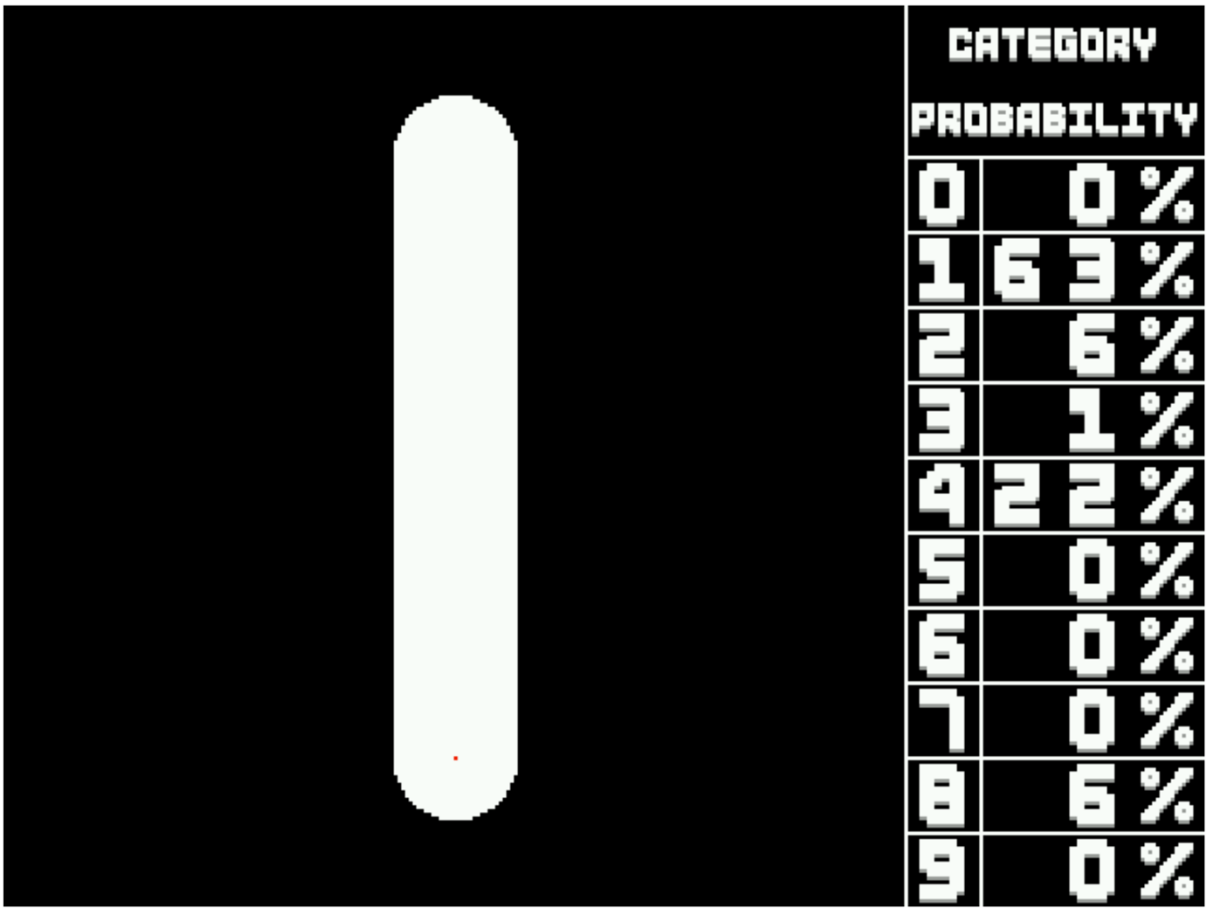

Fig. 3: user interface

Fig. 4 below shows the model architecture used for this handwritten digit classification task. It utilizes a similar architecture to the LeNet-5 model with small modifications to the activation functions from tanh to ReLU. We trained the model from scratch using the MNIST dataset of 28x28 pixel handwritten digit images. We then exported the weights and biases of each layer and imported them as double arrays in our C code for convolution and pooling operations during inference.

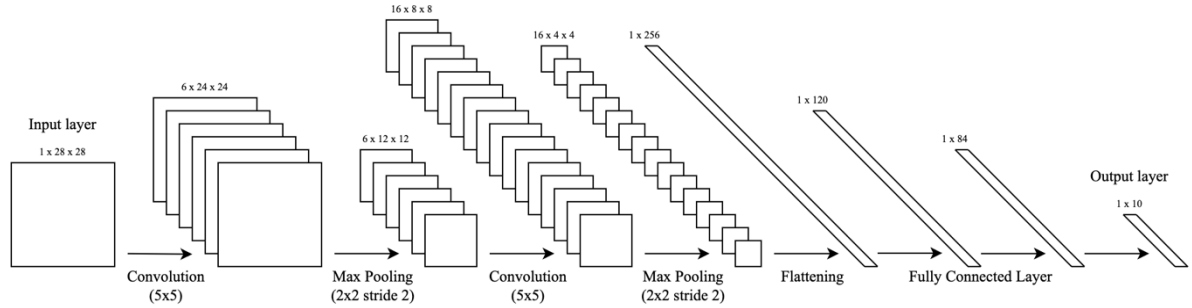

Fig. 4: CNN model architecture

Attribution Table:

	Train model (python)	Inference model (C)	Keyboard controls	Image resize	Image illustration	Hex display	Report	Total
Daniel	v	v		v	v		v	50%
Lily			v		v	v	v	50%