**Project Description:**

"Sudoku 2", A solver of sudoku puzzles that includes a neural network that can read a handwritten sudoku board.

**Competitive Analysis:**

There are many other sudoku solving services, like <https://www.sudokuwiki.org/sudoku.htm>, and <https://www.sudoku-solutions.com/>. However, most can not read handwritten sudoku boards. My project has a neural network along with image processing that allows it to read handwriting and can attempt to solve your handwritten boards for you as well.

**Structural Plan:**

My final project is predicted to be 4 python files.

The python files will be 1 for the neural network to recognize the numbers, 1 solving the sudoku puzzles, 1 for blob detection and for making each "blob" into something that the neural network can read, and 1 final runner file for the user interface so that the user can use the product.

**Algorithmic Plan:**

The two tricky parts of my program are creating a neural network and writing a program for blob detection. They were two areas that I didn't have any experience in and were difficult to understand and code.

The main algorithmic idea of the neural network is this: The program takes in some data and splits it into two sets, the training dataset (70%) and the testing dataset (30%). Then, it assigns each value of input (from the testing dataset) some weight and bias (kind of like m and b in y=mx+b where weight is m and bias is b). Then, it passes this info to another set of neurons that does the same thing. This happens once more and the final product is outputted. This output is a prediction. Then, loss (error) is calculated using the actual "answers" to the data from the training dataset for every layer of the network to see where the weights and biases were. Then, this loss is used to update the weights and biases in the network. Then, this process is repeated. After repeating several times, the weights and biases are fairly close to what they should be for the predictions to come relatively correctly. Then, to test this hypothesis, the testing data is sent through the network to get the testing accuracy. This number tells you how good your network is at predicting the answer when given some data.

Blob detection was the other majorly complex part of my project. In essence, my program takes an image, cuts it up into small pieces based on white space and then uses the image’s colour values (when converted to greyscale) to detect what is a number and where the number ends. Once the number is found, that area of the image is copied and the copy is made into an array of 1s and 0s to indicate where the writing is and where the blank page is. Then the numbers are scaled based on this array of 1s and 0s and resized to match the training data for my neural network. The numbers are then put into a list and send through my neural network to be identified.

**Timeline Plan:**

The neural network is working to the the best that I can make it. The blob detection works well. I have a way to check if the board given is valid. In the coming days, I will be making a solver for the game and UI so that the user experience is better.

**Version Control Plan:**

Versions of my project are saved on my Github (https://github.com/ShamikaD). In addition, I have been backing up my computer to a physical hard drive.

A screenshot of a social media post

Description automatically generated

**Module List:** numpy, matplotlib, and pandas