Title:

Sudoku Solver: A Machine Learning-Based Approach

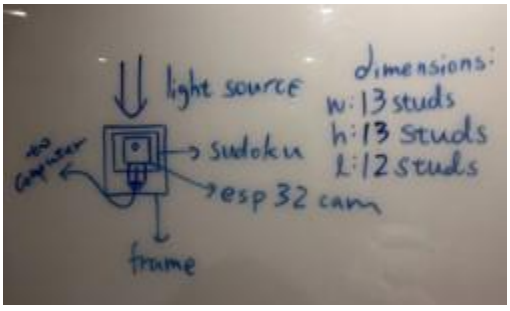
Problem Statement:

The objective is to develop a reliable Sudoku solver capable of recognizing handwritten or digital digits on a Sudoku grid, using a combination of hardware, machine learning models, and software. Challenges include achieving high accuracy in digit recognition and ensuring efficient integration of hardware and software components.

Method:

Prototype Design:

The prototype of the Sudoku Solver was constructed with Lego bricks, which allows for quick modifications and convenient measurements. The height and angle of the AI Thinker esp32 camera is kept constant due to the limitations of the machine learning algorithm. Initially, the design allows a light source to be placed at the opening. During testing, however, an iPad that emitted light was used for a sharper contrast between the number and the background. Openings on the side of the structure enable the user to access the two buttons on the side of the esp32 camera, one of which is the reset and is crucial when switching between programs.



My design was inspired by the lab professor and his students' suggestions for the initial version of the Sudoku Solver.

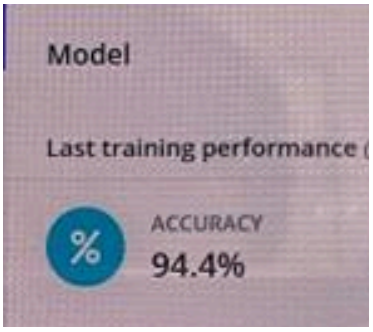
Camera view removed for privacy

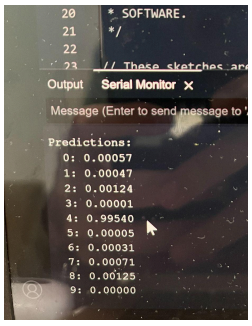
Using an example code for the esp32 cam in Arduino IDE that transmits data from the camera through wifi, I adjusted the angle, zoom, lighting, and other factors of the Sudoku image.

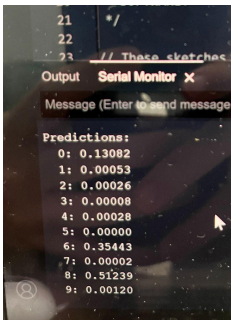
Machine Learning Implementation:

Stage 1:

Training began with uploading and labeling data from the MNIST database of handwritten digits. I used the website Edge Impulse's transfer learning model to train and test an initial model tasked with identifying a singular digit accurately and efficiently. After experimenting with the amount of data, training cycles, learning rates, and batch sizes, the model reached an accuracy or F1 score of roughly 94% without overfitting or underfitting. The model was then deployed in the form of an Arduino library and imported into Arduino IDE. Finally, an example program from the library was used to output the predictions for the digit every 7-8 seconds.



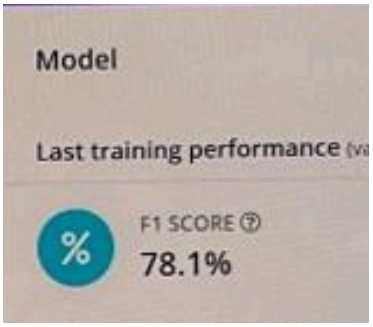


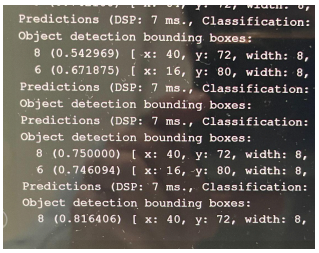


The output is formatted to display the digits followed by the probability of the digit written on the ipad being the given digit. During testing, I found that the digits have to be within a certain range of size and boldness to be properly detected by the camera. Moreover, the brightness of the iPad also has to be above a certain range to enhance the contrast. Nevertheless, after adjustment, the digit recognition system was relatively accurate.

Stage 2:

In order to scan the Sudoku in its entirety, the students at the lab suggested using the object detection model instead of transfer learning on Edge Impulse. However, several issues occurred, such as having to manually label each data point and not being able to achieve high accuracy or F1 score. While the labeling was eventually done with the help of AI generative tools, accuracy remained a problem due to the limited free access that the website provides. Thus, an alternative that I am currently exploring is transitioning to using Python and its machine-learning libraries to train models.





The output format for object detection consists of the digit it predicts, followed by the confidence level, the location, and the size of the numbers. There is a significant drop in accuracy compared to transfer learning, which is likely due to the more complex training for object detection.

Stage 3:
In response to the bottlenecks in Stage 2, I have worked with the students in the lab to produce a version of the number recognition system in Python in an attempt to increase accuracy. Additionally, since my Sudoku solver program was written in Python, it is much easier to fuse the two codes together. Currently, this stage is still in the development phase, and I have instead simplified the task to read a 3 by 3 grid of numbers and determine if it follows the rules of Sudoku for now. In the future, I plan to attempt different machine learning algorithms and increase the data size to make the number recognition system more accurate. I will likely upgrade the hardware tolerance and improve user accessibility as well to make it into a more useful product.

Result:

Current Achievements: Successfully implemented a digit recognition system with high accuracy for singular digits and sectional processing, which are parts of a preliminary Sudoku-solving product prototype.
Future Work: Developing Python-based solutions for enhanced accuracy and scalability, with plans to upgrade hardware and improve user accessibility.