**Computer Vision Boggle Solver**

By: Jordan Jarvis

Description: A program which uses a variety of Machine Learning and AI concepts to provide all possible words in a boggle board.

**Section 1:**

**Purpose**: The purpose of this project is to solve a boggle board using a combination of computer vision techniques, a neural network and threading; then it will return the results to the user. This will allow them to do very well in a game of boggle.

**Scope:** The scope of this project is to gain a better understanding of Machine Learning techniques. Included in this is the possibility to convert the algorithms used into something which can be sold on a mobile device app market.

**Overview:**

* + Why?

The reason for doing this is to learn about the processes involved in object detection, recognition, filtering, matrix math, and neural networks to improve the detection of objects in an image. In the end, boggle will be solved and all possible words will be provided to the user.

* + What?

As mentioned earlier, this program will do the seemingly simple task of playing a game. It will provide the user with the answers to a game of boggle. The libraries I will create will hopefully allow for recognition of characters in a grid. Potentially allowing for scrabble solving, sudoku solving, or other games which have letters and numbers in various orientations to be recognized and have a way to be solved using efficient algorithms.

* + Who?

This program will be targeted towards hiring managers and interviewers. The practical use of a boggle solver is low at best. The technical challenges to be solved will be a showcase of my ability to create a complex program which works well.

* + Where?

I will be using my computer and if enough time is available then it could have a GUI designed for it.

* + How?

The user will hold the boggle board inside of a square on their camera feed. After a few moments it will show the user all of the solutions it found.

* + When?

This project will be complete when it can view a boggle board and provide results. Additional polish can be applied to make it faster, more accurate, better interface, or more efficient.

Workflow:

Tasks and Schedule

* Research libraries involved and come up with a basic idea of how to segment the program. (Approx 6 hrs)
* Design interfaces, classes, and design documentation to a basic level (Approx 8 hours.)
* Learn how to show the images and results using a good layout and GUI (Approx 20 hours)
* Implement a board finder algorithm, (including filter algorithms, math to compute edges etc..) (Approx 25 hours)
* Implement a character finding and filtering algorithm (Approx 20 hours)
* Create data-sets for AI training using data from from character finding algorithm (Approx 11 hours)
* Design and train the Neural network algorithm (Approx 15 hours)
* Design boggle solving algorithm (Approx 10 hours)
* Connect trained network and image systems to boggle solving algorithm to print final results and show images (approx 15 hours)

**Section 2a:**

**Requirements:**

1.

**Requirement:** The program shall use a live camera feed.

**Success measurement:** The program can display a live feed of what it is seeing.

2.

**Requirement:** The program shall identify letters in various orientations from the camera feed when arranged in a grid.

**Success measurement:** A grid of size 4x4 with letters can be identified in a video feed and turned into a 4x4 matrix with an accuracy above 99%.

3.

**Requirement:** The program shall use multiple threads (multiprocessing in python).

**Success measurement**: Utilizing multiple processes allows the program to process frames faster by utilizing multiple cores. (should be measurable by allocating different number of cores)

4.

**Requirement:** The program shall identify all possible words in a given boggle board 4x4 grid.

**Success measurement**: Test cases will verify that all words are found automatically by comparing against boards which have already been solved.

5.

**Requirement:** The program shall create a 4x4 matrix from the camera feed and identify if what it is looking at is not a letter.

**Success measurement:** The matrix produced will identify when it does not have high (99%) confidence that what it is looking at is a character and return a unique value if that is the case.

6.

**Requirement:** This program shall use the camera feed or images to provide all possible words from a provided dictionary in a boggle board.

**Success measurement:** The program shall take the camera feed and produce all words in a boggle board.

**Section 2b:**

**Stretch Goals:**

1.

**Requirement:** The program shall show the words using a graphical interface.

**Success Measurement:** The words shall show on a GUI not in the terminal.

2.

**Requirement:** The program shall solve Sudoku puzzles.

**Success Measurement**: The same functionality shall exist but it shall have Sudoku and 9x9 grid detection as well.

**Section 3:**

**Design Overview of the Product**

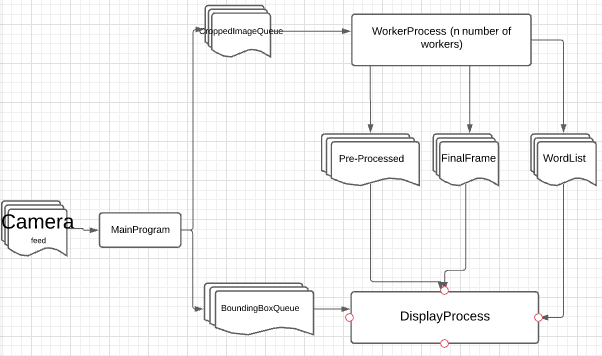
**Workflow:** The user opens the program, after which it shall spawn a GUI process and n number of worker processes. The main program will do a basic crop and put a bounding box on the original image . The bounding box image is put on a queue for the GUI process, and the basic crop is put on a queue for the worker processes, the GUI shall then show a camera feed with the bounding box. The camera feed shall ask the user to align the boggle board to a grid. Once aligned properly it shall feed the image into a variety of image processing techniques to remove useless data which would confuse the neural network. The processed image is split into n \* n smaller images which are fed in one at a time. The results shall either be a character, blank (0) or garbage (-1). If there are no blank or garbage values in the results then the results are converted back to a nxn grid of characters. The nxn grid is then passed in to a boggle solving algorithm, which returns a list of words on a queue to the display process.

**Resources:**

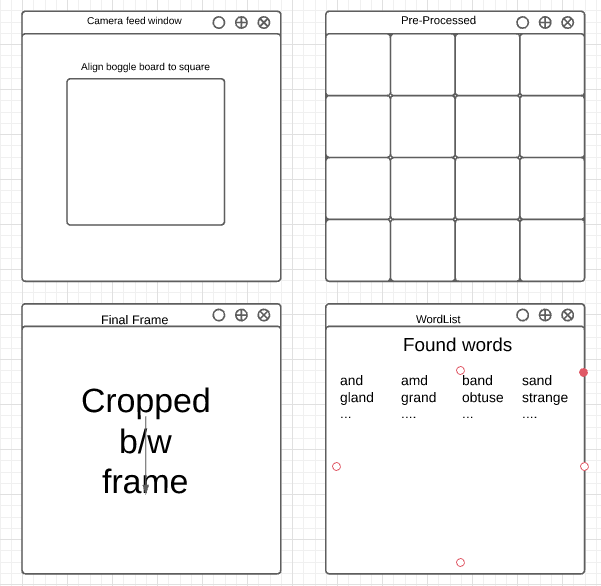
* Game of boggle
* Video Camera (USB or built in)
* GPU for accelerated image processing and recognition
* Computer to host the code on
* VSCode
* Python with various libraries installed
* TensorFlow
* Nvidia CUDA toolkit
* GitHub
* Lots of patience.

**Data at Rest:** Most data shall be processed on the fly. A file containing the words dictionary shall be in the root directory and may be swapped out with an end-line separated word list. The trained neural networks shall be stored as a descriptive folder or file in the root directory.

**Data State:**



**GUI:**



**Section 4:**

**Verification**

**Demo:** A screen recording shall be recorded with the program running from start to finish. The program is not complex on the front-end. The back end is much more complex.

**Testing:** Various test cases shall be developed.

**Test Case**: Main-Program passes frames to both queues of size 500x500. The queues have no more than 10 frames at a time, any extras are discarded.

**Test Case**: Display Process picks up and displays frame updates for each of the queues.

**Test Case:** Depending on implementation, tests shall be developed for each of the image manipulations to ensure they work correctly.

**Test Case:** The Neural Network takes in processed images and returns a matrix of the correct values.

**Test Case:** A sample boggle matrix of characters shall be passed in to the solver and a known number of words shall be returned.

**Test Case:** Neural Network testing shall return a rate above 99% accuracy rate for character identification.