Viewfinder

Kazi Jawad

Project Description

Viewfinder is an optical character recognition program that analyzes written math problems and solves for the variables. The user will have the option to upload an image, which can then be solved for through a trained neural network.

Competitive Analysis

Handwritten text recognition is a common project for people interested in deep learning and neural networks. There are also a large variety of open databases of handwritten words in the English language that can be used to train models.

However, a lot of programs will analyze the text and draw conclusions from the text, such as sentiment analysis. There are also programs that use the model to recognize the user's handwritten characters.

Viewfinder is unique because it pertains to a specific set of users. Viewfinder is targeted to academia for students and professors. The text that will be analyzed is a math problem that should be executed to a solution.

Structure

Viewfinder/
- data/ - Directory for training data set
- model/ - Directory for trained neural network
- src/ - Main directory for the program and neural network
  - classes/ - Directory of classes used by the neural network
  - main.py - Serves the program and different viewports
  - train.py - Trains the neural network
- examples/ - Directory of example images to test against

Algorithm

The core aspect of Viewfinder is the text recognition algorithm. The program will need to be able to analyze an image and detect text and interpret that text as characters.

To achieve the text recognition, I will need to use a multi-layer neural network that can decode handwritten text. This is achievable through Tensorflow and by applying a recurrent neural network on top of a convolutional neural network. The convolutional neural network will analyze the initial image and that information will feed to the recurrent neural network that will grab the text.

Since creating and training the model will be a large portion of the project, I decided to use a pretrained model to begin with as a placeholder. This

placeholder will allow me to create the interface and interactions between the user and program. The pretrained model will be replaced for the final deliverable.

Timeline

TP1 / November 20

- Implement text recognition on live video through OpenCV's dnn module with a pre existing model.
- Temporarily use a CLI for rapid prototyping.

TP2 / November 26

- Implement a custom text recognition model and train the data.
- Implement object detection of humans in live video to auto capture a screenshot.
- Transition from a CLI to a GUI.
- Create two separate routes for users to choose live video or images.
- Handle execution of code through python and display the output.

TP3 / December 05

- Provide a screen that allows people to edit the outputted text before execution.
- Integrate test functions against the output of the code.
- Add a caching/save function for different users.

Version Control

Git / Version Control System

- To keep track of local changes.

Github / Git Cloud Platform

- To backup git tree.

Modules

OpenCV / Computer Vision Library

- To be used for image processing and analysis.

Tensorflow / Machine Learning Platform

- To implement the text recognition model.

Numpy / Scientific Computing Library

- To be used for matrix manipulation in OpenCV.

Tkinter / Interface Toolkit

- To create the UI of the program.

TP2 Update

After the second meeting, we've decided to pivot the project to not be handwritten code analysis, but rather math function analysis. The stakeholder will use the app to upload an image of their handwritten function, which will then be analyzed by the neural network and solved for.

We also agreed the app can be generalized to take full suite of the neural network, since it's capable of reading most English language characters.

There will most likely be additional functionality, such as training the neural network itself and saving solutions.