Testing Considerations

sddec23:-02: Machine Learning Heterogeneous Computing

The plan tests a neural network system for pupil detection through unit testing (using training/testing data split), integration testing (between RPU, APU, DPU components), and system testing (measuring performance). Results show 2.54 pixel RMSE and 220 frames/second processing.

Strengths

1. Clear component integration testing
2. Specific performance metrics
3. Meaningful error measurements (2.54px compared to 35px pupil)

Weaknesses

1. Limited unit testing approach
2. No edge case testing
3. Underdeveloped acceptance testing

Strong Elements

1. **Automated Integration:** Automated drivers and database checks enhance accuracy.

2. **System Validation:** Combining automated and manual tests ensures reliability.

3. **Continuous Integration:** Github CI enables the to quickly issue identification and resolution.

Weak/Missing Elements

1. **Non-Functional Details:** Lacks clear performance metrics and timing criteria.

2. **Test Documentation:** Limited detailed documentation for clarity and reproducibility.

3. **Stress Testing:** Absence of explicit scalability and load testing strategies.

Our Approach:

Since our project is an extension on precious years, our focus is on optimizing and adapting the new idea of pipelining the algorithms already set in place. Our main goal is to divide the semantic segmentation algorithm alongside the other 3 algorithms that are already set in place, but in testing, the team wants to focus on the semantic segmentation algorithm. In doing so the team wants to ensure that the accuracy of the separated algorithm matches what is already set in place. In practice the team wants to start testing as soon as the team has working models, and make modifications as often as the team experiences issues early on.

This is prevalent due to the waterfall development process as the rest of the group cannot move forward unless this has been completely tested ahead of time which will be a recurring theme with any of this project's development as one portion of the project directly relies on one another. After dividing the algorithm the team will need to thoroughly test the thread as they handle the frames that are being fed into the pipelines. The threads accept an input that has the format of a cv::matt class and the team can test the thread’s functionality by

inputting a matrix equation in the same format. The goal would be to view each output of the thread as to what the team expects to be solved on paper and fed into a future thread to be put together in solving the overall “problem” the team is feeding the algorithm.

In addition to this the team also plans on including integration testing for all interfaces already included in the material the team is given to what the team develops, measure and compare past performance metrics to the solution the team comes up with, include error handling tests with any errors that may come up that the team don’t expect may break the running build on the board. In this entire process the team have and continue to plan on documenting test cases that the team comes up with to test these metrics and document the results that result. In compliance with our client, the team also involves them with any of our experimentation and testing to gain knowledge on whether or not the team should go forward with our ideas or navigate to other solutions as they are a valuable resource for the team.

Lastly, since our project heavily depends on work completed by previous teams both within and outside Iowa State University, we plan to rigorously test all provided materials to proactively identify and resolve any issues that might hinder our development process. To date, significant time has been spent waiting on the completion or correction of prior teams' tasks. In several instances, the team has needed to fix errors from previous teams even before their scheduled handoff.