Evaluation

My final solution meets almost all of the requirements in the requirements specification. The few originally outlined requirements which the solution does not meet were not met deliberately, as the designs and intentions of the project were changed as development occurred. These requirements and the reasons why they have not been met are detailed later in the evaluation. The final solution also meets the technical requirements by including recursion and arrays of objects. Overall, I would deem this project a success, although ideally more time would be spent on it if possible.

Requirements Specification

The only requirements that were not met are the following:

* The requirements specification originally stated that the project should finished by the 21st of March. This was not achieved, due to the fact that the deadline for project hand-in was extended into April.
* User Requirement 3: Project should have a brief explanation of how neural nets work. This requirement had actually been satisfied throughout the vast majority of development, as can be seen in the high fidelity prototype. However, this was removed after some end-user feedback in order to include explanatory text, thus further providing User Requirement 1: intuitive interface. In my opinion, sacrificing requirement 3 in order to more effectively satisfy requirement 1 was the right choice to satisfy the majority of users.
* Functional Requirement 7: Creatures able to eat other creatures as well as food pellets. This function was intentionally removed during implementation, as after some research and experimentation, it was decided that the feature would actually hinder the creature’s abilities to learn.
* Functional Requirement 8: Neural nets stored in text files. This feature was also intentionally removed during implementation as it was realised it was not in fact required, and would hinder how intuitive the user interface was.

Through both end-user and on-going testing I have discovered that the program runs smoothly and is robust enough to deal with anything a user may choose to try to do. This was achieved through a clear user interface which only allowed tem to do things that the program is intended to do. Additionally, many systems were setup to print error messages if anything were to go wrong, so that the errors can be quickly identified and their sources easily found.

Testing

Component testing was carried out during the development of the solution, in order to test isolated components as they were created, and to catch bugs and errors early on in development before they became integrated with the whole program.

Integrative testing was also carried out at regular intervals during development in order to ensure that all components are able to work in tandem with each other.

Both integrative and component testing were very successful, allowing development to move on at a steady pace as I became confident that any already existing components worked correctly. It also did expose a few errors early on which were addressed.

Final testing was comprised of usability, acceptance, and functionality testing.

Functionality testing was carried out on the solution as a whole in order to test how smooth it was, and whether it met the functional requirements. This was generally a success, ignoring the functional requirements specified above.

Usability testing was carried out in order to find out how intuitive the user interface was. Potential end-users were asked to use the program without any supervision and then asked to fill out a survey. The majority of the users surveyed were computer science students as they were the target audience, however a couple of novice users were also surveyed in order to get less biased results. Overall, this was deemed a great success as all the users found the program interface intuitive and relatively easy to use, even those who deemed themselves novice users.

Acceptance testing was carried out to find out how interested users were in the program, and how satisfying the experience was for them overall. Overall, this was mostly regarded as a success, as the majority of users reported that they enjoyed the experience, and those that  
didn’t also reported that they were not interested in machine learning to begin with.

End-user testing also provided a great number of potential features that are planned for future development, as detailed later in this report.

Personal Performance

During development of this project a record of progress was used to keep track of what was being worked on and note reflections about difficulties encountered and any changes to the design of the program. This was very useful, as it helped me stay on track with what I was working on, and made writing commit messages when using Git much easier.

Overall, I believe I could have more successful with timekeeping as I did end up having to work on the project much more often towards the end of development than at the start, instead of spreading it out more evenly throughout development. As the deadline was approaching it also became clear that I would not be able to implement many of the additional features which I had designated to only be dealt with once every other aspect of the project was complete. This was fine, as it had always been intended that these features may not be implemented, however if I were to work on a project of this magnitude again I would wish to take much more care with timekeeping.

I think the choice of an object-oriented programming language to develop this program in was an excellent choice, as it massively increased the efficiency of coding the project. I believe if a non-object-oriented language had been chosen it would have taken significantly longer to develop this project.

Part of the reason I chose to undertake