System Design Document

For

EasyNN/AI

David Fadini

Nathan Rose

Liam Kehoe

Nathan Foster

Jack Nguyen

Raymond (Sky) Kwasneski

Version/Author	Date
1.0 / Fadini, Rose, Kehoe, Foster, Nguyen, Kwasneki	2/16/2021
1.1 / Fadini, Rose, Kehoe, Foster, Nguyen, Kwasneki	3/18/2021
1.2 / Fadini, Rose, Kehoe, Foster, Nguyen, Kwasneki	4/6/2021

TABLE OF CONTENT

SYSTEM DESIGN DOCUMENT

Overview
The System Design Document describes the system requirements, operating environment, system and subsystem architecture, files and database design, input formats, output layouts, human-machine interfaces, detailed design, processing logic, and external interfaces.

1 INTRODUCTION

1.1 Purpose and Scope

EasyNN: The SDD will lay out the general structure for the EasyNN and where responsibilities lay.

EasyAI: The SDD will lay out the general look and structure that the programmers can develop using the predetermined Django structure. It will also layout the look and feel of the final product to model from.

1.2 Project Executive Summary

1.2.1 System Overview

This section describes the system in narrative form using non-technical terms. It should provide a high-level system architecture diagram showing a subsystem break out of the system, if applicable. The high-level system architecture or subsystem diagrams should, if applicable, show interfaces to external systems. Supply a high-level context diagram for the system and subsystems, if applicable. Refer to the requirements traceability matrix (RTM) in the Functional Requirements Document (FRD), to identify the allocation of the functional requirements into this design document.

EasyNN:

EasyAI:

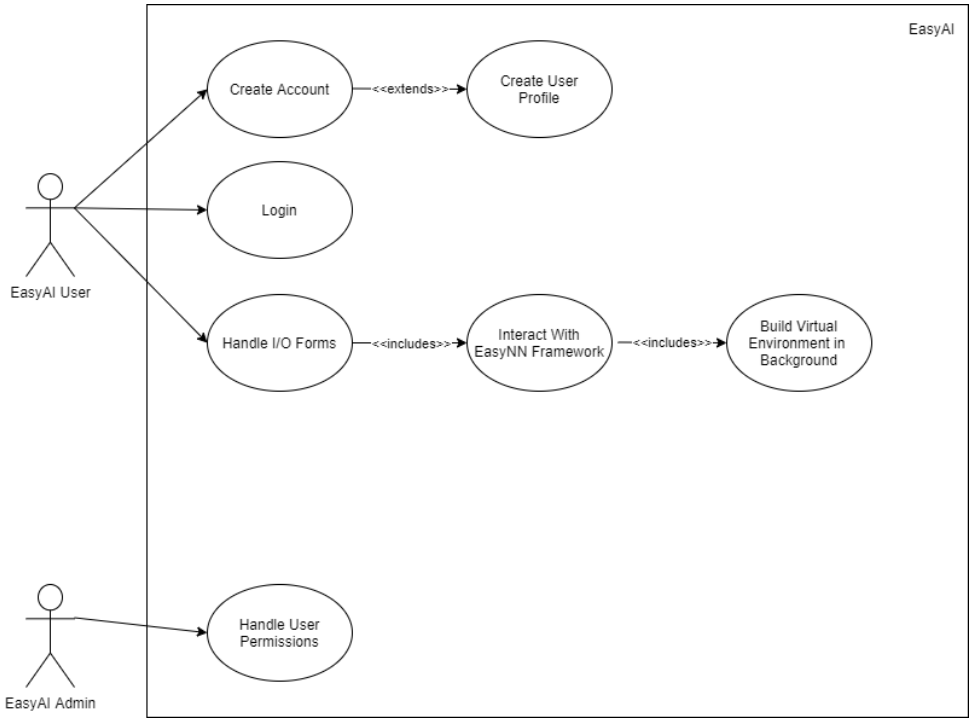

Figure 1: Use Case Diagram for EasyAI System

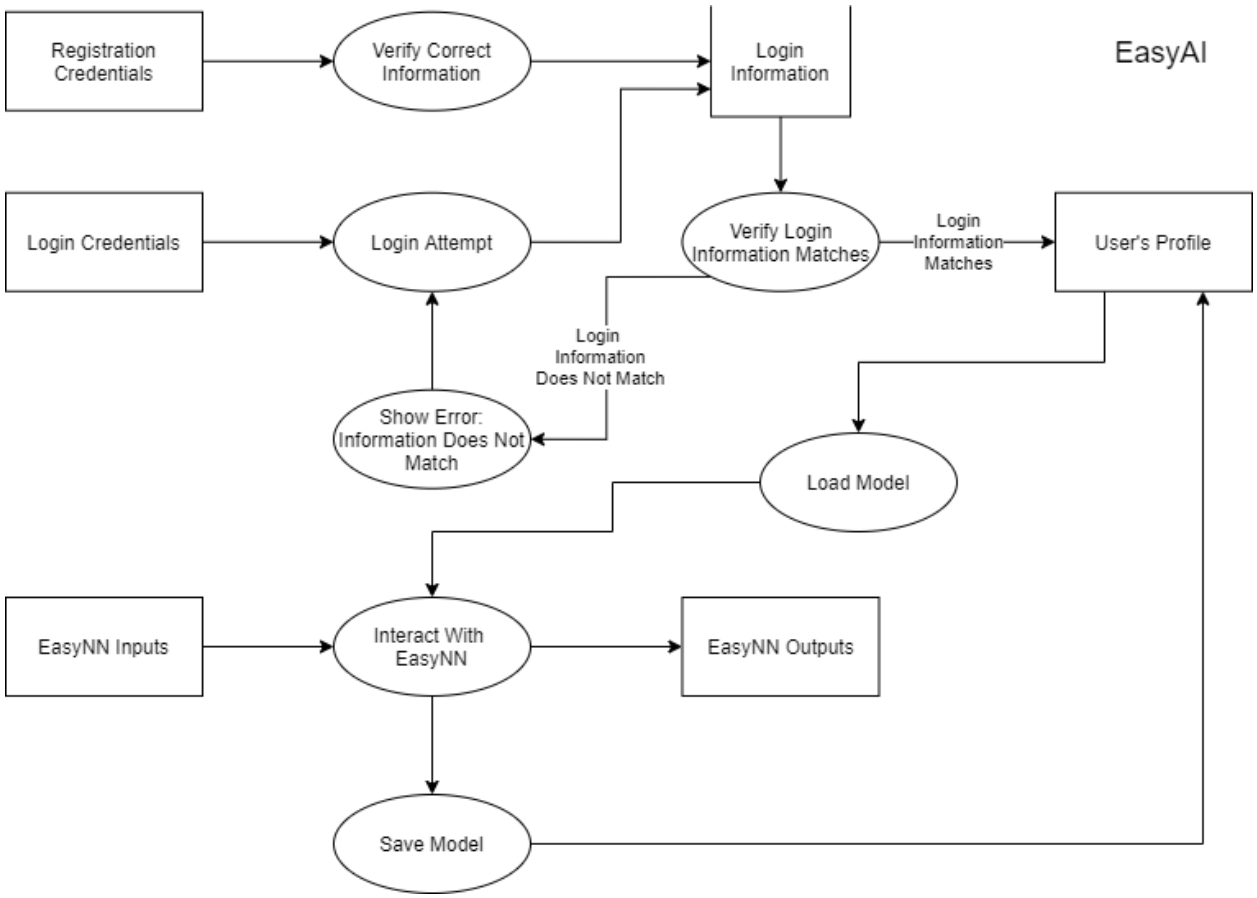

Figure 2: System Overview of EasyNN

1.2.2 Design Constraints

This section describes any constraints in the system design (reference any trade-off analyses conducted such as resource use versus productivity, or conflicts with other systems) and includes any assumptions made by the project team in developing the system design.

EasyNN: The design is constrained by Python since it is a high-level language. This limits the efficiency of the software.

EasyAI: The design is constrained by the Django framework since it is the core component of the project. It forces very specific modularity that, if not followed perfectly, will now allow EasyAI to function at all. Software design can only be as flexible as Django will allow.
The web server must also be stable and fast at its core, due to the nature of the EasyNN algorithm being implemented. If the server runs poorly as a static website, then it will throttle (if it will even allow) the EasyNN code.

1.2.3 Future Contingencies

This section describes any contingencies that might arise in the design of the system that may change the development direction. Possibilities include a lack of interface agreements with outside agencies or unstable architectures at the time this document is produced. Address any possible workarounds or alternative plans.

EasyNN: If EasyNN proves to be too difficult for users, then EasyNN should be made more user-friendly by improving the documentation or software design.

EasyAI: If the EasyNN portion takes up too many resources, then the cosmetics of the website will have to be reduced and simplified. The direct plan is to start EasyAI simple and then implement more once the core of EasyNN is set up with smooth operation for the user.

1.3 Document Organization

This document is designed to give the reader an idea of the system design. The following sections will provide information on what the product does, limitations, interactions, interfaces, hardware and software designs, and security.

1.4 Project References

This section provides a bibliography of key project references and deliverables that have been produced before this point.

EasyGA was a previous Python package designed by some of the same team members. The EasyGA framework allows users to implement genetic algorithms. The EasyGA framework will be used in conjunction with the EasyAI framework to allow genetic algorithms to be run on a server. The EasyGA framework will be used in conjunction with

the EasyNN framework to allow neuroevolutionary training.

1.5 Glossary

Supply a glossary of all terms and abbreviations used in this document. If the glossary is several pages in length, it may be included as an appendix.

EasyNN:

1. Model: A representation of another system.
2. NN: An artificial neural network, which is a simplified computational model of the human brain, loosely based on the idea of passing information between several nodes to get an output.

EasyAI:

Django: The web server framework used for EasyAI

2 SYSTEM ARCHITECTURE

In this section, describe the system and/or subsystem(s) architecture for the project. References to external entities should be minimal, as they will be described in detail in Section 6, External Interfaces.

2.1 System Hardware Architecture

In this section, describe the overall system hardware and organization. Include a list of hardware components (with a brief description of each item) and diagrams showing the connectivity between the components. If appropriate, use subsections to address each subsystem.

EasyNN:

EasyAI:

2.2 System Software Architecture

In this section, describe the overall system software and organization. Include a list of software modules (this could include functions, subroutines, or classes), computer languages, and programming computer-aided software engineering tools (with a brief description of the function of each item). Use structured organization diagrams/object-oriented diagrams that show the various segmentation levels down to the lowest level. All features on the diagrams should have reference numbers and names. Include a narrative that expands on and enhances the understanding of the functional breakdown. If appropriate, use subsections to address each module.

EasyNN:

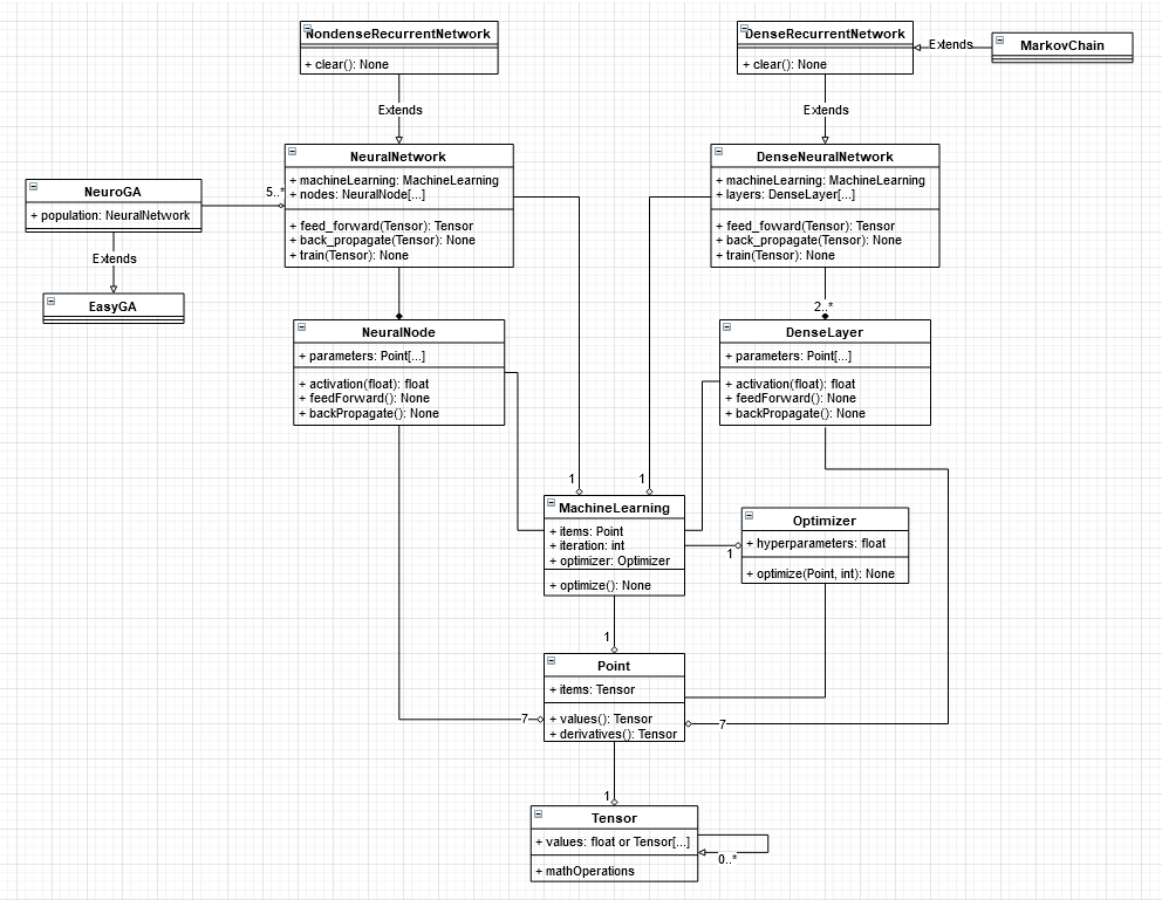

Figure 3. EasyNN UML Diagram

EasyAI: The software architecture of the Django framework is still being learned and expanded upon. This will be updated with the full diagram once more information on the design is known.

2.3 Internal Communications Architecture

In this section, describe the overall communications within the system; for example, LANs, buses, etc. Include the communications architecture(s) being implemented, such as X.25, Token Ring, etc. Provide a diagram depicting the communications path(s) between the system and subsystem modules. If appropriate, use subsections to address each architecture being employed.

Note: The diagrams should map to the FRD context diagrams.

EasyNN:

EasyAI: The software architecture of the Django framework is still being learned and expanded upon. This will be updated with the full diagram once more information on the design is known.

3 HUMAN-MACHINE INTERFACE

This section provides the detailed design of the system and subsystem inputs and outputs relative to the user/operator. Any additional information may be added to this section and can be organized according to whatever structure best presents the operator input and output designs. Depending on the particular nature of the project, it may be appropriate to repeat these sections at both the subsystem and design module levels. Additional information may be added to the subsections if the suggested lists are inadequate to describe the project inputs and outputs.

3.1 Inputs

This section is a description of the input media used by the operator for providing information to the system; show mapping to the high-level data flows described in Section 1.2.1, System Overview. For example, data entry screens, optical character readers, bar scanners, etc. If appropriate, the input record types, file structures, and database structures provided in Section 3, File and Database Design, may be referenced. Include data element definitions, or refer to the data dictionary.

Provide the layout of all input data screens or graphical user interfaces (GUTs) (for example, windows). Provide a graphic representation of each interface. Define all data elements associated with each screen or GUI, or reference the data dictionary.

This section should contain edit criteria for the data elements, including specific values, range of values, mandatory/optional, alphanumeric values, and length. Also, address data entry controls to prevent edit bypassing.

Discuss the miscellaneous messages associated with operator inputs, including the following:

- Copies of the form(s) if the input data are keyed or scanned for data entry from printed forms
- Description of any access restrictions or security considerations
- Each transaction name, code, and definition, if the system is a transaction-based processing system

EasyNN:

EasyAI:
The users will be inputting information through the EasyAI website. This information and data get passed through to the EasyNN system and processed. The EasyAI website is used as a hub, where users can save and train their models in a user-friendly GUI.

3.2 Outputs

This section describes the system output design relative to the user/operator; shows mapping to the high-level data flows described in Section 1.2.1. System outputs include reports, data display screens, and GUIs, query results, etc. The output files are described in

Section 3 and may be referenced in this section. The following should be provided, if appropriate:

- Identification of codes and names for reports and data display screens
- Description of report and screen contents (provide a graphic representation of each layout and define all data elements associated with the layout or reference the data dictionary)
- Description of the purpose of the output, including identification of the primary users
- Report distribution requirements, if any (include frequency for periodic reports)
- Description of any access restrictions or security considerations

EasyNN:

EasyAI: The GUI is very simplistic at the moment. The idea is to add an import section for users to import data into for the EasyNN algorithm to operate on, and the website will output the result of the training (this will vary based on which algorithm the user is looking for).

4 DETAILED DESIGN

This section provides the information needed for a system development team to build and integrate the hardware components, code and integrate the software modules, and interconnect the hardware and software segments into a functional product. Additionally, this section addresses the detailed procedures for combining separate COTS packages into a single system. Every detailed requirement should map back to the FRD, and the mapping should be presented in an update to the RTM and include the RTM as an appendix to this design document.

4.1 Hardware Detailed Design

EasyNN:
EasyNN has no hardware components required of it since it is a python package that is downloaded from PyPI's servers.

EasyAI:
EasyAI will be running on a server through an AWS system or a server that runs inside of my home. The distinction will be made based on the demand of the algorithms on the system. If the cost of running the AWS system gets too high due to computational processing then switching to the local server will make the most sense since the only additional cost will be the higher electricity bill.

Figure 4. Locally operated server

Hardware specifications:

- R720 Server:
  - 256GB of ram
  - 12 core/24 thread
- Senology
- Switch
- Backup Power

Software Detailed Design

A software module is the lowest level of design granularity in the system. Depending on the software development approach, there may be one or more modules per system. This section should provide enough detailed information about logic and data necessary to completely write source code for all modules in the system (and/or integrate COTS software programs).

If there are many modules or if the module documentation is extensive, place it in an appendix or reference a separate document. Add additional diagrams and information, if necessary, to describe each module, its functionality, and its hierarchy. Industry-standard module specification practices should be followed. Include the following information in the detailed module designs:

- A narrative description of each module, its function(s), the conditions under which it is used (called or scheduled for execution), the overall processing, logic, interfaces to

other modules, interfaces to external systems, security requirements, etc.; explain any algorithms used by the module in detail
- For COTS packages, specify any call routines or bridging programs to integrate the package with the system and/or other COTS packages (for example, Dynamic Link Libraries)
- Data elements, record structures, and file structures associated with module input and output
- Graphical representation of the module processing, logic, flow of control, and algorithms, using an accepted diagramming approach (for example, structure charts, action diagrams, flowcharts, etc.)
- Data entry and data output graphics; define or reference associated data elements; if the project is large and complex or if the detailed module designs will be incorporated into a separate document, then it may be appropriate to repeat the screen information in this section
- Report layout

EasyNN:
The EasyNN system is divided into two main groups of classes, which are then extended from. On the left, the NeuralNetwork and NeuralNode classes are related and extended by the NondenseRecurrentNetwork class. On the right, the DenseNeuralNetwork and DenseLayer classes are related and extended by the DenseRecurrentNetwork class, which is further extended by the MarkovChain class. The NeuralNetwork is further involved in the NeuroGA class, which extends the GA class from EasyGA. The Tensor class is a recursive data structure, which is formatted by the Point class for other usage. The MachineLearning class is used in the Network classes and holds an Optimizer and a Point. The MachineLearning class' Point's Tensor is then split across the NeuralNode and DenseLayer classes as Points with Tensors.

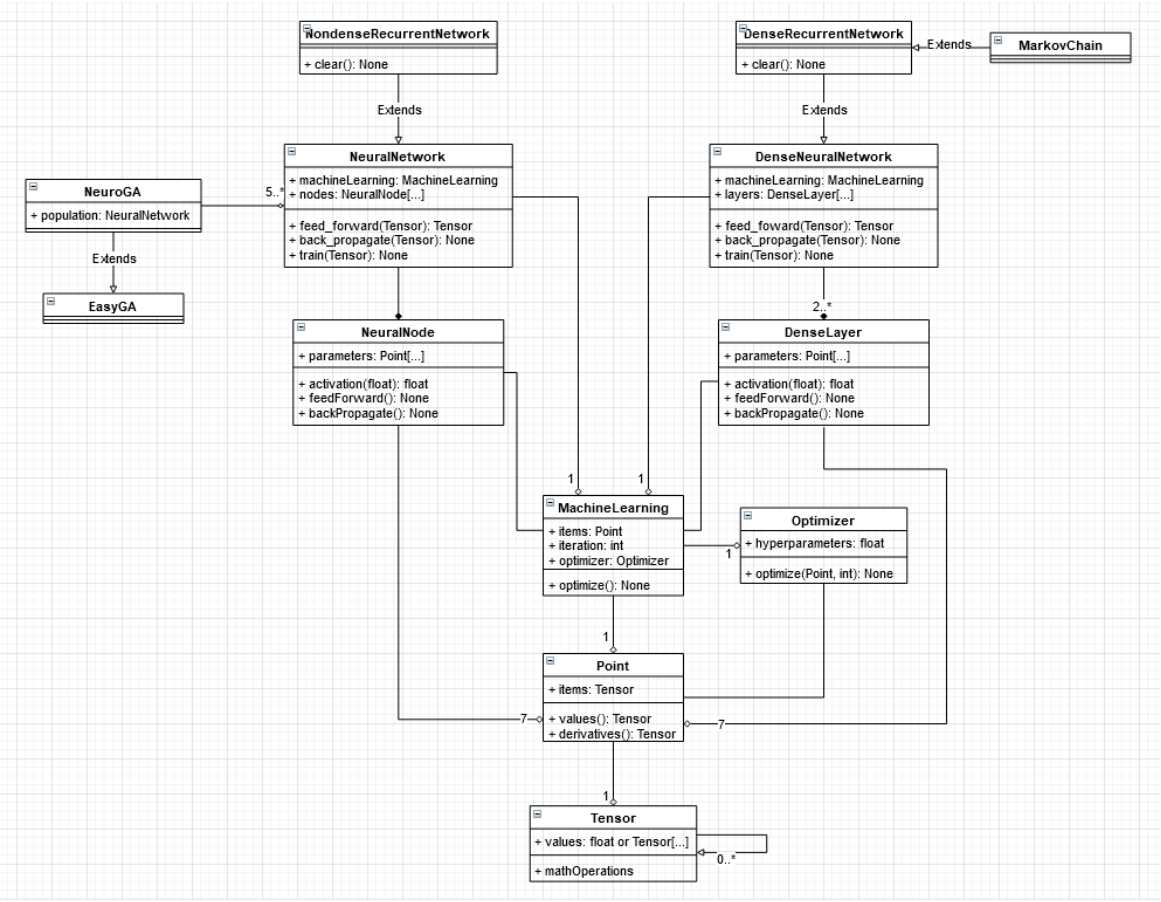

Figure 5. EasyNN UML Diagram

EasyAI: The baseline modules are the urls.py, views.py, and the HTML page templates for each of the included pages. The urls.py file contains the held URLs for the different pages of the website and how to handle the different directions given in the web browser. The views.py contains the information on what is going to be presented on the different pages and passes the appropriate pages to the HTML files to show up on the website.

4.2 Internal Communications Detailed Design

If the system includes more than one component there may be a requirement for internal communications to exchange information, provide commands, or support input/output functions. This section should provide enough detailed information about the communication requirements to correctly build and/or procure the communications components for the system. Include the following information in the detailed designs (as appropriate):

- The number of servers and clients to be included on each area network
- Specifications for bus timing requirements and bus control
- Format(s) for data being exchanged between components
- Graphical representation of the connectivity between components, showing the direction of data flow (if applicable), and approximate distances between

components; information should provide enough detail to support the procurement of hardware to complete the installation at a given location

- LAN topology

EasyNN:

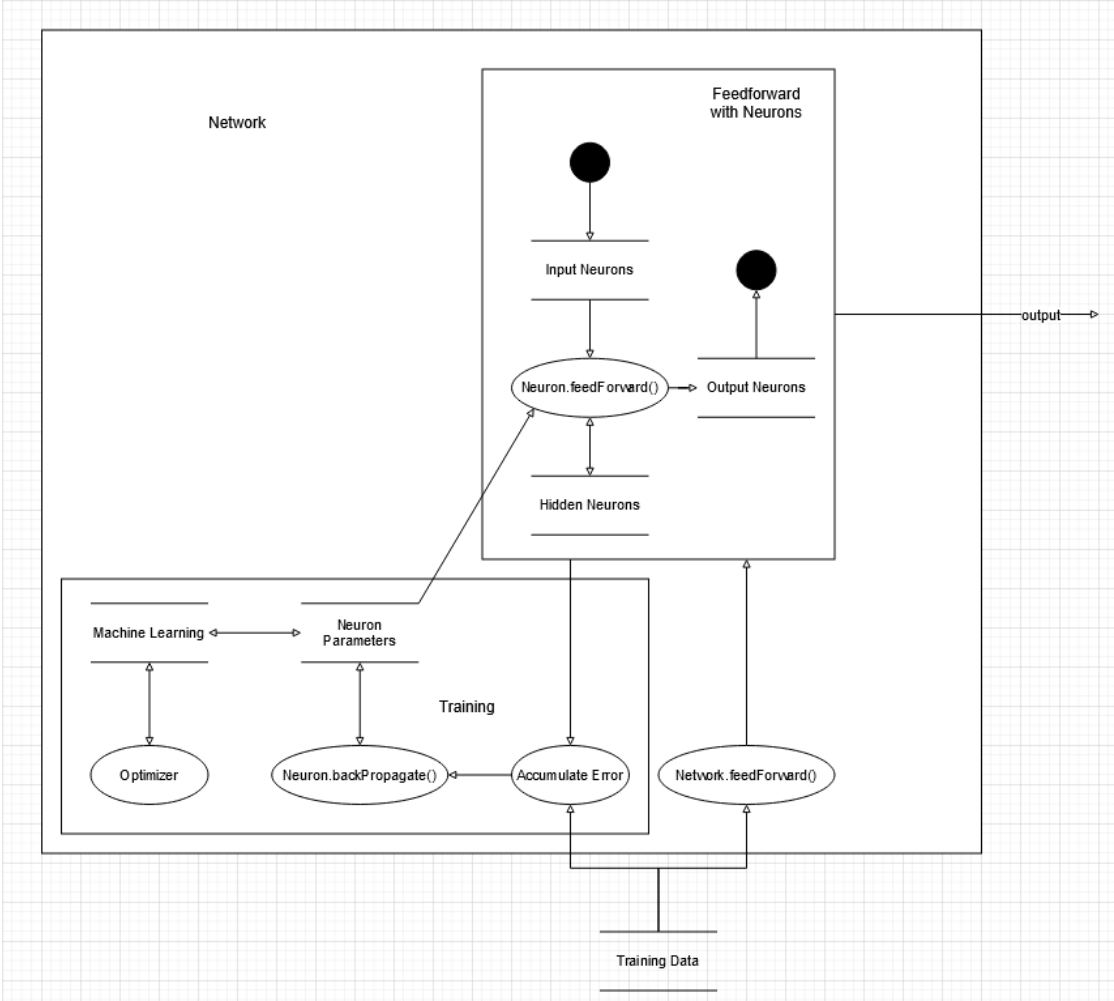

Figure 6.

There are two main sections of data flow in EasyNN. The feedforward process is composed of passing input into the system and producing an output. When the input is passed into a network, it passes it along to the neurons. These neurons then pass the information along with each other and into their respective points for specialized storage. Finally, the output is produced from the system. The backward training involves looping the feedforward process to accumulate error using given training data. The backward training then passes this information through the neurons in the reversed order. This then modifies hidden neuron parameters, which are tied to the machine learning class. Finally, the values are optimized using an optimizer.

EasyAI: Everything in the web server is built together and the flow of information goes through all of the predetermined paths of the Django framework, which has already been

tested rigorously by its developers. The EasyAI team is simply following the path created by those developers in the creation of the server.

5 EXTERNAL INTERFACES

External systems are any systems that are not within the scope of the system under development, regardless of whether the other systems are managed by the State or another agency. In this section, describe the electronic interface(s) between this system and each of the other systems and/or subsystem(s), emphasizing the point of view of the system being developed.

5.1 Interface Architecture

In this section, describe the interface(s) between the system being developed and other systems; for example, batch transfers, queries, etc. Include the interface architecture(s) being implemented, such as wide area networks, gateways, etc. Provide a diagram depicting the communications path(s) between this system and each of the other systems, which should map to the context diagrams in Section 1.2.1. If appropriate, use subsections to address each interface being implemented.

EasyNN:

EasyAI: This is all handled with the Django automatic setup framework, so little changes are required for the developers to make it work properly.

5.2 Interface Detailed Design

For each system that provides information exchange with the system under development, there is a requirement for rules governing the interface. This section should provide enough detailed information about the interface requirements to correctly format, transmit, and/or receive data across the interface. Include the following information in the detailed design for each interface (as appropriate):

- The data format requirements; if there is a need to reformat data before they are transmitted or after incoming data is received, tools and/or methods for the reformat process should be defined
- Specifications for hand-shaking protocols between the two systems; include the content and format of the information to be included in the handshake messages, the timing for exchanging these messages, and the steps to be taken when errors are identified
- Format(s) for error reports exchanged between the systems; should address the disposition of error reports; for example, retained in a file, sent to a printer, flag/alarm sent to the operator, etc.
- Graphical representation of the connectivity between systems, showing the direction of data flow
- Query and response descriptions

If a formal Interface Control Document (ICD) exists for a given interface, the information can be copied, or the ICD can be referenced in this section.

EasyNN:

EasyAI: This does not pertain too much to controlling a web server and setting up the HTML/CSS of the web pages. An HTML form will be used to take in the data from the user on the information of the EasyGA/EasyNN information at first, and the form will take in integers for the evolution steps.

6 SYSTEM INTEGRITY CONTROLS

Sensitive systems use the information for which the loss, misuse, modification of, or unauthorized access to that information could affect the conduct of State programs or the privacy to which individuals are entitled.

Developers of sensitive State systems are required to develop specifications for the following minimum levels of control:

- Internal security to restrict access of critical data items to only those access types required by users
- Audit procedures to meet the control, reporting, and retention period requirements for operational and management reports
- Application audit trails to dynamically audit retrieval access to designated critical data
- Standard Tables to be used or requested for validating data fields
- Verification processes for additions, deletions, or updates of critical data

Ability to identify all audit information by user identification, network terminal identification, date, time, and data accessed or changed.

EasyNN: EasyNN is an installable package, meaning multiple users are going to be installing and messing with some of the usages of the different components. There are no major/critical data on any portion of the package that can be stolen.

EasyAI: The website is going to be as secure as Django allows, with its internal data being inaccessible to the average user and a password-protected admin page.