SOFTWARE REQUIREMENTS SPECIFICATIONS FOR NEUROMATIC

Submitted by:

Connor Arnold

Henry Gridley

Griffin Knipe

Daniel Vosburg

Date:

07/22/18

1. INTRODUCTION

1.1 Purpose of this Document

This document describes the functions and performance features of the Neuromatic neural network application. This application will provide a visual medium for the construction and training of neural networks.

1.2 Scope of the Development Project

The purpose of this product is to provide a visual tool to design and test neural network configurations as an alternative to writing code by hand. This will be accomplished through a graphical user interface where the user can place neural network layers on a canvas. The user will be able to edit the configuration of neural network layers, and specify the data on which the network will be trained, to create a working model for future use. This product can be extended to include more complex neural network layers, such as convolutional layers. This product can also be extended to provide a service, similar to Amazon Web Services, that allows a user to utilize remote processing power to train neural networks.

1.3 Definitions, Acronyms, and Abbreviations

1.3.1 Definitions

- **Activation Function**: A function that converts an input signal to an output signal within a neuron.
- **Canvas**: The workspace, presented by the application graphical user interface, where the user will implement a neural network design.
- **Feature**: An individual property being observed in a neural network.
- **Generate neural network**: Create a python script using Keras, which can be used to create a model.
- **Hidden Layer**: A network component, whose inputs and outputs are not exposed beyond the network.
- **Input Layer**: A network component, whose input is exposed outside of the network and output is not.
- **Layer**: A collection of artificial neurons, at a specific depth, within a neural network.
- **Model**: Mathematical representation of the parameters within a neural network. Makes predictions on data.
- **Neural Network:** A classification system emulating the computational behavior of the human brain.
- **Neuron**: A function contained in the neural network which is trained to output a specific value when given an input data feature.

- **Output Layer**: A network component, whose output is exposed outside of the network and input is not.
- **Train neural network**: Use training data to modify parameters within the network and create a model.
- **Training Data**: A processed collection of data points for which the desired output is already known. This data will be used to train the neural network.

1.3.2 Acronyms and Abbreviations

- **API**: Application Programming Interface
- **CPU**: Central Processing Unit
- **CSV**: Comma Separated Values
- **GB**: Gigabyte
- **GUI**: Graphical User Interface
- **MB:** Megabyte
- **NN**: Neural Network
- **RAM**: Random Access Memory
- **SRS**: Software Requirements Specifications

1.4 References

[1] Docs.python.org. (2018). *24.1. Tkinter — Python interface to Tcl/Tk — Python 2.7.15 documentation*. [online] Available at: https://docs.python.org/2/library/tkinter.html [Accessed 19 Jul. 2018].
[2] Stack Overflow. (2018). *Stack Overflow - Where Developers Learn, Share, & Build Careers*. [online] Available at: https://stackoverflow.com/ [Accessed 19 Jul. 2018].
[3] TensorFlow. (2018). *API Documentation | TensorFlow*. [online] Available at: https://www.tensorflow.org/api_docs/ [Accessed 19 Jul. 2018].

1.5 Overview of Document

The remainder of this document consists of four sections: General Description, Specific Requirements, Other Requirements, and Appendix.

Section 2, General Description, describes the application's intended user and lists the features that will address the user's needs. This section also describes the bounds guiding the application's development, including constraints, assumptions, dependencies, and guidelines.

Sections 3, Specific Requirements, details the functional requirements, listed in Section 2, and introduces the application's nonfunctional requirements. This section also describes the external interface requirements and the necessary environment to run this application.

Section 4, Other Requirements, is not applicable to this application. All necessary requirements are covered by the rest of this document.

Section 5, Appendix, contains GUI mockups and block diagrams that are referenced throughout the document. These figures will help the reader to understand the functionality of the product.

2. GENERAL DESCRIPTION

2.1 User Characteristics

The user is expected to have an academic or professional programming background or an equivalent level of experience. The user is expected to have training data, which will include data points and labels that can be processed by a NN. The user is expected to lack the NN expertise necessary to implement a NN given a task's time constraint. Therefore, the user has a need for a tool that will assist in the development of a NN by rectifying the user's knowledge and time limitations. The GUI will address this need by abstracting the coding aspect of NN design to a visual representation. Instead of writing code, the GUI can be used to design a high level schematic of a NN.

2.2 Product Perspective

2.2.1 Product Relationships

Neuromatic is a standalone product using the Keras API to:

- Provide a library of NN layers.
- Construct a sequential network.
- Train a model on the user's data.

2.2.2 External Interfaces

The product will interface with the user through a GUI.

2.3 Overview of Functional Requirements

2.3.1 Open New Canvas

The user will be able to open a blank canvas.

2.3.2 Increase/Decrease Slot Amount

The user will be able to increase or decrease the number of slots for layers on the canvas.

2.3.3 Add/Remove Layers

The user will be able to add or remove layers from the canvas.

2.3.4 Modify Component Properties

The user will be able to modify layer properties.

2.3.5 View Application Status

The user will be able to view the application status (string) from the GUI.

2.3.6 Clear Canvas

The user will be able to clear all layers from the canvas.

2.3.7 Save Canvas

The user will be able to save the current canvas' properties.

2.3.8 Load Canvas

The user will be able to open previously saved canvas properties.

2.3.9 Generate Python Script

The user will be able to generate a Python script representing the NN logic created in the canvas.

2.3.10 Stop Progress

The user will be able to stop the current process without the crashing the application.

2.3.11 Input Training Data Path

The user will be able to specify a path to the training data.

2.3.12 Specify Model Location

The user will be able to specify the file location of the created model.

2.3.13 Train Model

The user will be able to use the neural network to train a model.

2.3.14 Create New Canvas

The user will be able to create a new canvas.

2.3.15 Resizing the Application's Window

The user will be able to resize the window.

2.3.16 Minimizing the Application's Window

The user will be able to minimize the window.

2.3.17 Closing the Application

The user will be able to close the window.

2.4 Overview of Data Requirements

This product takes two types of input from the user. The first type of input consists of the current NN properties. NN properties include layer size and activation function for each NN layer, as well as the canvas properties such as output file name. Appendix D shows how these properties are passed from the GUI, to the control script, and then arrives at the "Generate Neural Network Code" backend script or the "Train Model" backend script.

The second type of input is training data used to train the network. This will be data the user has preprocessed and will feed into the network to generate a model. The generated model is one of the system outputs, which the user can use in other applications to make predictions on new data.

The other system output is the generated NN source code. Due to the potential hardware limitations of the user's machine, the user will be able to take the source code and train the network outside of this application.

2.5 General Constraints, Assumptions, Dependencies, Guidelines

2.5.1 Constraints

Training data must be in CSV format. Data sets cannot exceed a size of 30 MB. The product will be a desktop application compatible with Unix-based operating systems. The system running the application must have have at least 4 GB of RAM and at least a dual-core CPU faster than 2.0 GHz. The system constraints will affect the user at runtime and may limit the NN size, or the amount of data that can be used to train the model. System processing power and network training time have an inverse relationship.

2.5.2 Assumptions

It is assumed that, the user will have a basic understanding of how neural networks function and that the user has an academic or professional programming background or an equivalent level of experience.

2.5.3 Dependencies

The application requires the installation of Python 3.6.5, TensorFlow, and Keras.

2.5.4 Guidelines

The source code for this program will be found on GitHub. The user will follow the instructions in the README file to install this application all required Python modules. The README will also contain guidelines, with required inputs and outputs, for creating additional network components not included in this product.

2.6 User View of Product Use

When the user launches the application, a GUI will open and present a new canvas as shown in Figure 1. A new canvas will be composed of three empty slots. Each slot represents a space where the user can place a NN layer. To place a NN layer, the user will select a layer type from the "New Components" sidebar at the right of the GUI. Once a layer is placed, the user may remove a layer by selecting the layer and then selecting the trash can in the bottom left corner.

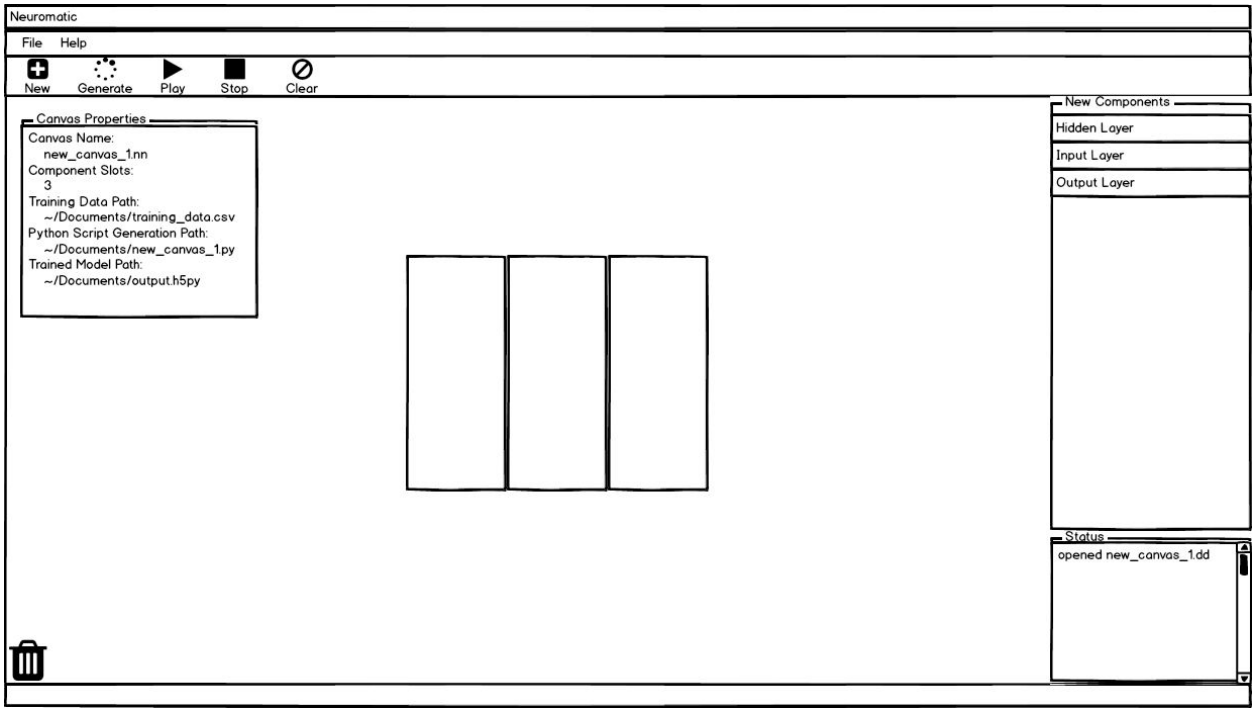


Figure 1. Wireframe representation of a blank canvas.

Appendix A shows a canvas with its three slots filled with an Input Layer block, a Hidden Layer block, and an Output Layer block. The face of each block lists properties that will affect the

behavior of the model. The user will be able to select the block and edit its properties using the window shown in Appendix B.

The “Canvas Properties” box, located at the top left of the canvas in Figure 1, presents the user with adjustable parameters. Selecting and opening the box will present the window shown in Appendix C. There, the user can set the canvas name, the number of slots, and the file path for the training data. The output file path for the generated Python script and the trained model can also be set in that window.

The “Status” text box at the bottom right of the GUI will display progress updates and errors generated by the application. The status data will be passed between application modules as shown in Appendix D. The top bar menu will consist of “File” dropdown and a “Help” button. The “File” dropdown will contain buttons allowing the user to save the current canvas or open a previously saved canvas. The “Help” button will open the ReadMe file which will provide application use guidelines.

Just below the top bar, there will be a bar of buttons. The user will be able to click the “New” button to open a new canvas. Once the user is finished editing the canvas, the user can then click the “Generate” button to generate the NN Python code. The user will be able to then click “Run” to train the model and output the trained model file.

3. SPECIFIC REQUIREMENTS

3.1 External Interface Requirements

The GUI will be the user’s interface to all NN parameters such that the user will not have to hand manipulate code while designing a NN. Along the right side of the canvas will be individual layers which the user can place into the canvas slots. Once a layer is placed on the canvas, the user can adjust the layer’s properties using the dialogue box shown in Appendix B. The user can modify the output of the program using the dialogue box shown in Appendix C.

This program requires Python 3.6.5 to be installed and running on the user’s computer. This program is built on top of Keras and Tensorflow, both of which can be installed following the set up instructions found in the README. This program will support Unix operating systems. A computer mouse and keyboard is required to perform basic functionality in the program.

3.2 Detailed Description of Functional Requirements

3.2.1 Open New Canvas (2.3.1)

Purpose: The user will be able to open a blank canvas.
Inputs to the Component: Mouse click.
Processing:

1. Check if the "New Canvas" button has been clicked by the user.
2. Check if a canvas is already open.
3. If a canvas is already open, the application will prompt the user to save the current canvas, delete the current canvas, or cancel the request for a blank canvas.
4. If the user chooses to save the canvas or there was no canvas open, a new canvas will be displayed by the GUI.
5. Otherwise, the current canvas will continue to be displayed.

Outputs: If the user chooses to save the canvas or there was no canvas open, a new canvas will be displayed by the GUI. Otherwise, the current canvas will continue to be displayed.

3.2.2 Increase/Decrease Slot Amount (2.3.2)

Purpose: The user will be able to increase or decrease the number of slots for layers on the canvas. Shown in Appendix E as Edit Project Properties.
Inputs to the Component: Mouse clicks, and keyboard input.
Processing:

1. Check that the user clicked edit project properties.
2. Project properties pop-up is displayed.
3. User inputs the desired number of slots.
4. If the number of slots input is less than 1 or greater than 10, notify the user of invalid input.
5. If desired slot number is less than number of slots that currently exist, remove slots from right side of canvas. If layers exists in the removed slots, the layers are deleted.
6. Else add empty layers to the right of the canvas.

Outputs: Return an invalid input error if unsuccessful. Display a new canvas view with desired number of slots if successful.

3.2.3 Add/Remove Layer (2.3.3)

Purpose: The user will be able to add or remove neural network layers to or from the canvas. Shown in Appendix E as Add/Remove Block.
Inputs to the Component: Mouse clicks.

Processing:

1. Check that a new layer has been clicked.If step one is true, check that a slot has also been clicked.
2. Check if user-selected slot is empty.
3. If the slot is free, the new neural network component will be placed in the desired slot.
4. Else, the layer will not be placed and the GUI will display an error explaining why the new component could not be placed.

Outputs: A new neural network layer will be placed in the desired slot if the conditions above are satisfied. Otherwise, the component will not be placed and the GUI will display an error explaining why the new layer could not be placed.

3.2.4 Modify Layer Properties (2.3.4)

Purpose: The user will be able to modify the properties of each component, such as the number of neurons. Shown in Appendix E as Edit Block.

Inputs to the Component: Mouse clicks, Field modifications describing different parameters for the selected component.

Processing:

1. User clicks edit layer properties
2. Component properties window is displayed and user inputs desired parameter.
3. Check that the desired parameters will allow for a valid neural network.
4. If these conditions are true, change the parameter to the desired value.
5. Else, return an error message.

Output: Change the parameter to the desired value if successful. Otherwise, the status text box will display an error explaining why the parameter could not be changed.

3.2.5 View Application Status (2.3.5)

Purpose: The user will be able to view the application status.

Inputs to the Component: Logger output piped in, shown in Appendix D.

Processing:

1. Receive status or error from the application logger.
2. If the new status is different from the most recently displayed status, then print the new status to the GUI status text box.
3. Else, do not output new status.

Output: The GUI will display the new status if successful. If unsuccessful, the GUI will not display the new status.

3.2.6 Clear Canvas (2.3.6)

Purpose: The user will be able to remove all components currently on the canvas.

Inputs to the Component: Mouse clicks.

Processing:

1. Check if the clear canvas button has been clicked.
2. If the application is processing training data, print an error message in status text box.
3. Else, clear all layers from the canvas slots.

Output: The GUI will display the all empty slots, unless the generation or training script is running, in which case an error status will be displayed

3.2.7 Save Canvas (2.3.7)

Purpose: The user will be able to save the current canvas components and settings to a user-specified file path to return to later.
Inputs to the Component: Mouse clicks.
Processing:

1. User selects file->save.
2. File browser is opened.
3. User selects file location.
4. If selected file path is empty, save the file.
5. If the selected file path is not empty, ask the user if the existing file should be overwritten.
6. If the user selects yes, overwrite the file.
7. If the user selects no, return to step 2.
8. If condition 4 is not true, save the file.

Output: The file describing the current layout of the canvas is saved to the user selected location if the that location is empty or the user chooses to overwrite the existing file.

3.2.8 Open Canvas (2.3.8)

Purpose: The user will be able to open a previously saved canvas
Inputs to the Component: Mouse clicks.
Processing:

1. User selects file->open.
2. File browser is opened.
3. User selects file path to desired canvas configuration.
4. Check that the file is a canvas configuration.
5. If not, notify the user that the file is not a valid canvas configuration.
6. Else, program parses the canvas data from the file.
7. Program overwrites current canvas settings to match those found in the file.

Output: If the selected file path is a valid canvas configuration, the canvas is converted to the state of the canvas defined in the user selected file.

3.2.9 Generate Python Script (2.3.9)

Purpose: The user will be able to generate the neural network code found on the canvas. Shown in Appendix E as Generate Neural Network.
Inputs to the Component: Mouse clicks. Neural network design on canvas.
Processing:

1. User clicks generate button.
2. Canvas is checked for valid data inputs.
3. If canvas data is valid, proceed. Else, throw error in status box and stop process.
4. Canvas NN configuration data from GUI is parsed and sent to the code generation script.
5. Code generation script modifies the base NN code by adding lines based on the canvas components.
6. NN code is saved to the path specified in the canvas properties box shown in Appendix C.
7. Status displays that the code has been generated.

Output: The Python code which can create the neural network designed on the canvas, saved as a Python file at the user-specified location.

3.2.10 Stop Progress (2.3.10)

Purpose: The user will be able to cancel the training of a model without crashing the system.
Inputs to the Component: Mouse click on stop button.
Processing:

1. User clicks the stop button.
2. If the model is being trained, the training will be stopped and the products of the partial training, such as the model file, will be deleted. Status message displayed.

Output: No model will be created and the application will be idle. Cancel message is displayed in status box.

3.2.11 Input Training Data Path (2.3.11)

Purpose: The user will be able to input the training data file path to the program.
Inputs to the Component: Mouse clicks.
Processing:

1. User clicks the Canvas Properties box to open the dialogue box shown in Appendix C.
2. User clicks Browse next to training data file field.
3. File browser is opened.
4. User selects path to file from the file browser.
5. Check if selected folder is valid. If valid format display the chosen file name in the project properties box. Else, throw error message in status box and discard changes.

Output: Display the chosen file name in the project properties box.

3.2.12 Specify Model Location (2.3.12)

Purpose: The user will be able to specify the file location of the created model.
Inputs to the Component: Mouse clicks.
Processing:

1. User clicks the Canvas Properties Dialogue box to open the dialogue box shown in Appendix C.
2. User clicks Browse next to the Trained Model Path data field.
3. User selects path to file from the file browser.
4. Change the Trained model file path displayed on the Canvas Properties box.

Output: Display the chosen file name on the GUI.

3.2.13 Train Model (2.3.13)

Purpose: The user will be able to train the model using the specified training data. Shown in Appendix E as Train Neural Network.
Inputs to the Component: Button click.
Processing:

1. Check that neural network code has been created and is in correct location.
2. If so continue, else throw status error.
3. A thread will run the train NN script.
4. The train NN script will run the generated Python code and output to the user defined model file location.

Output: The NN model file will be created at the user specified location.

3.2.14 Create New Canvas (2.3.14)

Purpose: The user will be able to create a new blank model.
Inputs to the Component: Button click.
Processing:

1. User clicks “New” button.
2. Warning appears to confirm deletion of current canvas.
3. If confirmed continue, else cancel process.
4. Set canvas settings back to default.

Output: The canvas will look like its default view seen in Figure 1.

3.2.15 Resizing the Application’s Window (2.3.15)

Purpose: The user will be able to resize the window.
Inputs to the Component: Mouse click and drag.
Processing:

1. The user adjusts the size of the GUI.

2. The program will adjust the size and/or position of the buttons to keep position ratios intact.

Output: The application window resized. The GUI widget sizes change to accomodate the window size.

3.2.16 Minimizing the Application's Window (2.3.16)

Purpose: The user will be able to minimize the window.
Inputs to the Component: Button Click.
Processing: The window will minimize into the tray.
Output: The application will no longer be visible.

3.2.17 Closing the Application (2.3.17)

Purpose: The user will be able to close the window.
Inputs to the Component: Button Click.
Processing: The application will stop all script activity and close the window.
Output: The application will be closed.

3.3 Performance Requirements

1. The training data for the product will be in CSV format, and will have a maximum file size limit of 30 MB.
2. The time between processing each data point will be tracked. If the average time exceeds a threshold calculated based on the number of data points and the number of epochs, the system will timeout to prevent training data with many features from consuming too much of the system's RAM.

3.4 Quality Attributes

3.4.1 Reliability

1. When the "train" button is pressed, the product will train the NN on the training data, or display an error message if:
   a. The data file is in the incorrect format
   b. The data does not fit the input format specified by the user
   c. A timeout interval is exceeded between processing each data point, indicating the user's machine does not contain powerful enough hardware to train the network.

3.4.2 Extendability

1. Developers will be able to extend the functionality of the product by adding new component types.

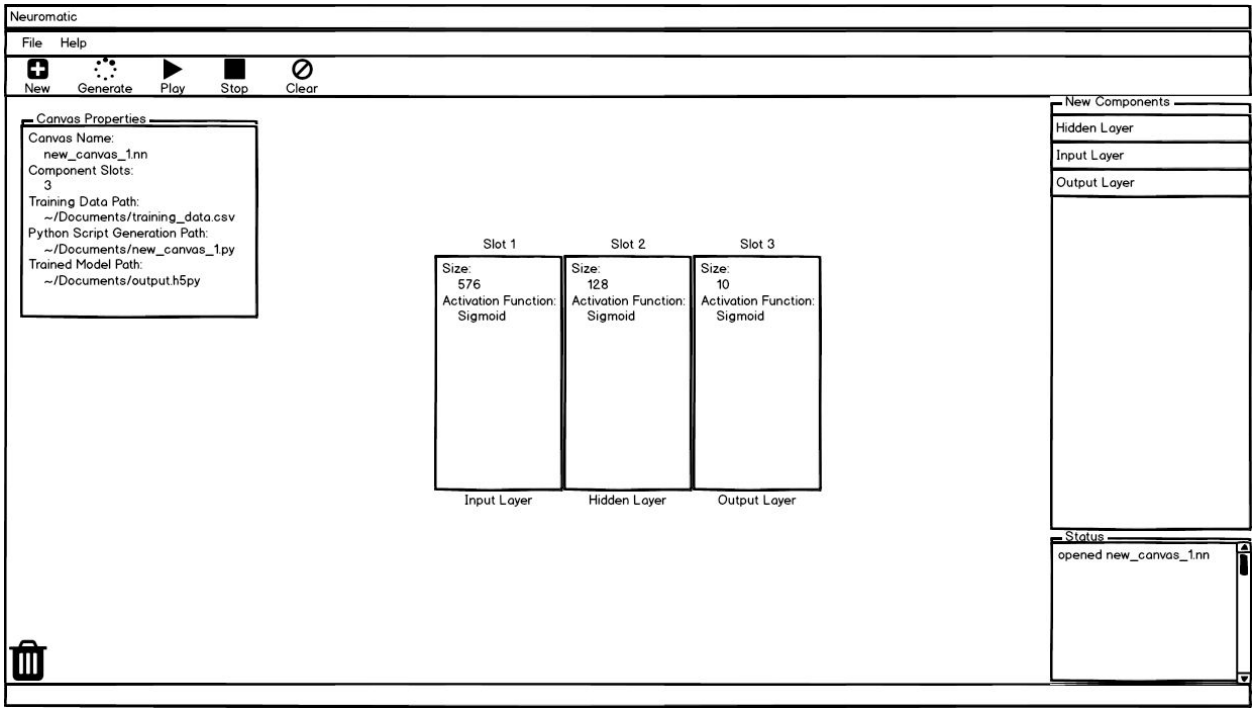
2. Developers will be able to clone the GitHub repo and submit pull requests with changes.

4 OTHER REQUIREMENTS

Not applicable

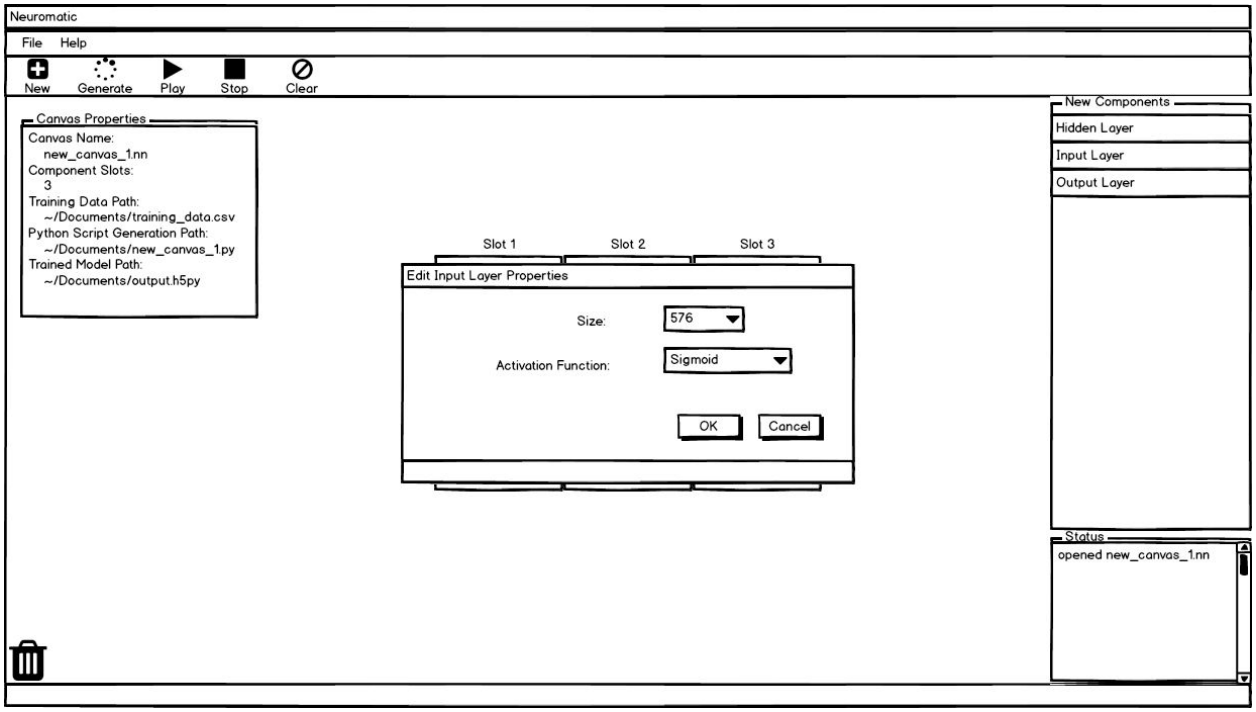
5 APPENDIX

Appendix A



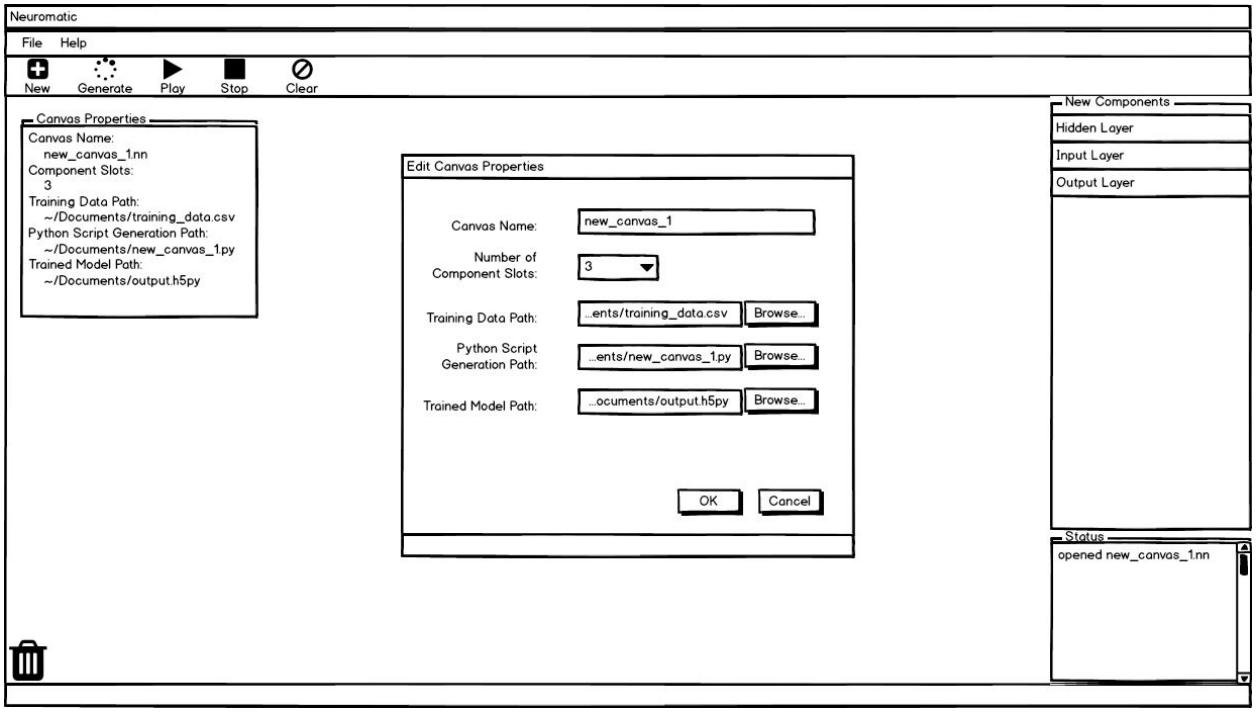
Wireframe representation of an example neural network.

Appendix B



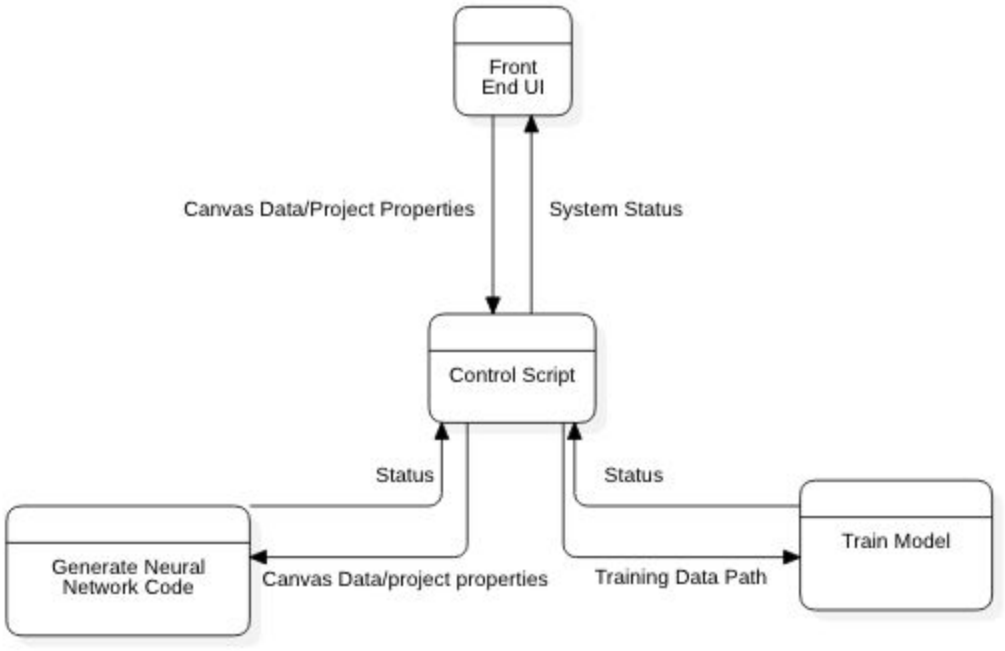
Wireframe representation of editing layer properties.

Appendix C



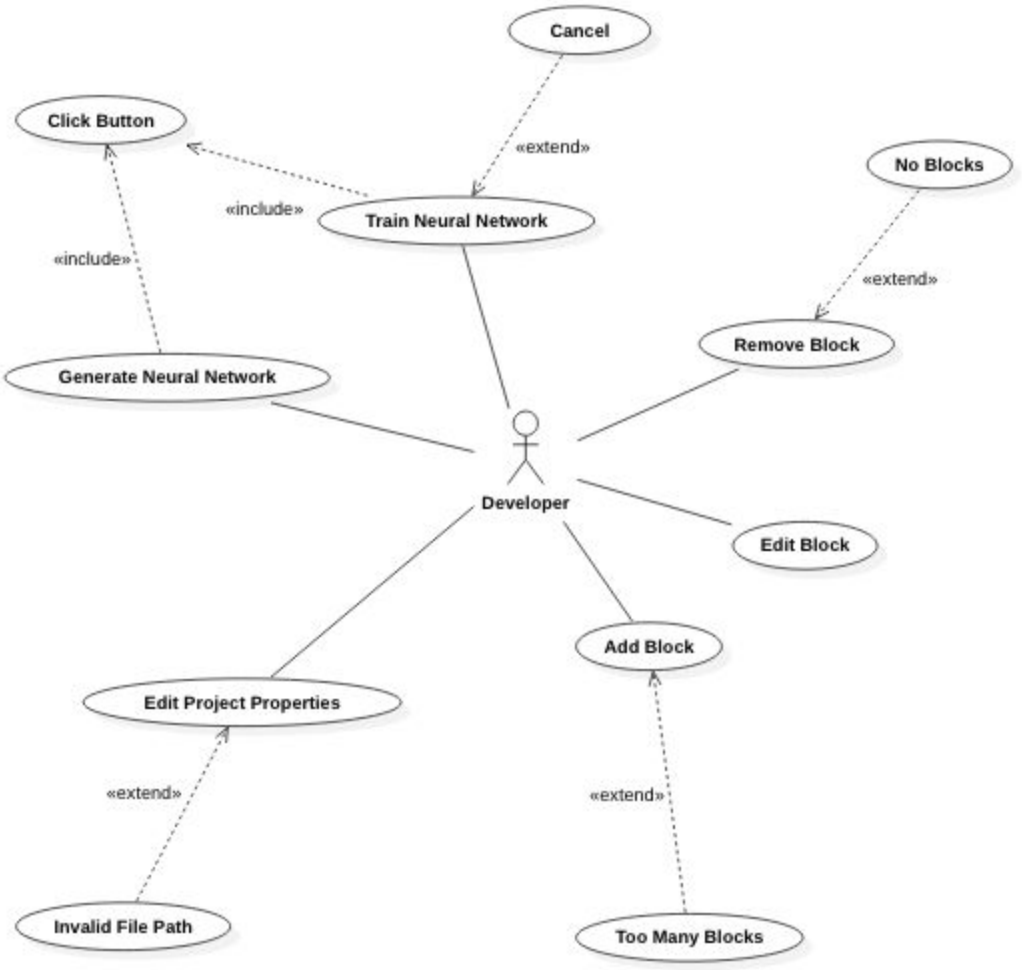
Wireframe representation of editing canvas properties.

Appendix D



Application Data Flow Diagram

Appendix E



Application Use Case Diagram