TEST PLAN FOR NEUROMATIC

Submitted by:

Connor Arnold

Henry Gridley

Griffin Knipe

Daniel Vosburg

Date:

08/14/18

1 TEST CASE DEFINITION AND EXECUTION

1.1 System Initialization Test

Application test - Test that the application opens correctly.

1.1.1 Requirements to be tested

1. The user will be able to open a blank canvas.
2. The user will be able to view the application status (string) from the GUI.

1.1.2 Oracle

- Success
  - The GUI displays a new canvas.
  - The status box displays a string, stating that the application has been started.
- Failure
  - The GUI does not initialize.
  - The application crashes.
  - The correct status string is not displayed in the status box.

1.1.3 Expected Test Result

The program window will open to the correct size with all components in place and the welcome message will be displayed in the status box.

1.1.4 Test Execution

STEP #	STEP DESCRIPTION	(INPUT) DATA	RESULT
1	Run the application via command line	Python3 command input	Program will launch opening up the window with welcome message displayed in status box

1.1.5 Result Summary

Test Successful. The application ran as expected after running the start command via the command prompt.

1.2 System Termination Test

Application test - Test that the application closes correctly.

1.2.1 Requirements to be tested

1. The user will be able to close the window

1.2.2 Oracle

- Success
    - The application window closes
- Failure
    - The application freezes
    - The application crashes

1.2.3 Expected Test Result

At any point, the user will click the close button on the window. This will close the application without crashing or freezing.

1.2.4 Test Execution

STEP #	STEP DESCRIPTION	(INPUT) DATA	RESULT
1	Click the close button on the application window	Mouse click	Application closes

1.2.5 Result Summary

Test Successful. The application was terminated and the window closed after the button was clicked

1.3 Application Management Test

Application test - Test that the application's window size can be modified

1.3.1 Requirements to be tested

1. The user will be able to resize the application's window
2. The user will be able to minimize the application's window

1.3.2 Oracle

- Success
    - The window's dimensions will be changed by dragging the border
    - The window will be hidden in the taskbar by pressing the minimize button and is put back by clicking the application in the taskbar
    - The window will be maximized by clicking the maximize button.
- Failure
    - The window will not be hidden in the taskbar on launch
    - The window will resize to the dimensions specified by the user when dragging the border
    - The program crashes
    - The application will operate as intended after maximizing

1.3.3 Expected Test Result

A double-headed arrow will appear when hovering over the application window borders. Clicking and dragging in any direction will change the window's dimensions. Clicking the minimize button at the top of the window will hide it in the taskbar and it will not be shown until the user clicks the application in the taskbar. Clicking the maximize button will expand the window to the edges of the screen. The application will operate correctly after maximizing.

1.3.4 Test Execution

STEP #	STEP DESCRIPTION	(INPUT) DATA	RESULT
1	Modify window size	Mouse click and drag on edge	Window size changes
2	Minimize window	Mouse click on minimize button	Window minimizes

3	Re-open window	Mouse click on minimized window	Window returns to screen
4	Maximize Window	Mouse click the Maximize button	Window expands to the screen edges
5	Drag hidden layer onto empty slot	Drag and drop hidden layer	On Linux and Mac OS 10.13.4 , the layer is successfully placed on the canvas. On Mac OS 10.13.6, the input layer is placed on the canvas.

1.3.5 Result Summary

Test partially successful. The window size was manipulated by dragging the window corner, minimized, and re-opened. The window was successfully maximized and operated as expected after the window was maximized, on Linux and Mac OS 10.13.4. However, on Mac OS 10.13.6, the application operated incorrectly after the window was maximized, with the cursor operating with offset input. This is shown in step 5 of Section 1.3.4. A layer other than the layer chosen by the user was placed on the canvas.

1.4 Canvas Management Test

User interface test - Ensure that the users actions affect the canvas and layer properties as intended.

1.4.1 Requirements to be tested

1. The user will be able to increase or decrease the number of slots for layers on the canvas.
2. The user will be able to clear all layers from the canvas.
3. The user will be able to create a new canvas.

1.4.2 Oracle

- Success
    - The correct number of slots are displayed, according to the Canvas Properties Box.
    - The layer types and properties are removed from all slots when the “Clear Slots” button is clicked.
    - A default canvas is displayed when the “New Canvas” button is clicked.

- Failure
  - The canvas is displayed in a state other than the state input by the user.
  - The number of slots listed in the Canvas Properties Box does not match the number of slots displayed on the canvas.
  - Slots displayed on the canvas cannot open the Layer Properties Box or interact with the drag and drop interface.

1.4.3 Expected Test Result

The GUI will display a canvas matching the number of slots specified in the canvas properties box. The canvas will change to the default state when the "New Canvas" button is clicked. All layers will be removed from canvas slots when the "Clear Slots" button is clicked. Canvas slots will retain their UI functionality while they exist on the canvas.

1.4.4 Test Execution

STEP #	**STEP DESCRIPTION**	**(INPUT) DATA**	**RESULT**
1	Raise component_slots to 10.	Text input via Canvas Properties Box	7 empty slots are added to the right of the canvas for a total of 10 slots.
2	Lower component_slots to 5.	Text input via Canvas Properties Box	5 slots are removed from the right of the canvas.
3	Drag a layer type onto an empty slot.	Mouse click and drag	The layer is added to the empty slot.
4	Click the "Clear Slots" button	Mouse Click	The layer is removed from the canvas. The number of slots remains the same.
5	Click the "New Canvas" button.	Mouse Click	The default canvas is displayed, consisting of 3 empty slots.

1.4.5 Result Summary

Test Successful. The number of canvas slots always matched the number specified in the Canvas Properties Box. Clicking the “Clear Slots” button removed the added layers from their slots. Clicking the “New” button set the canvas to its default state.

1.5 Layer Management Test

User interface test - Ensure that the user can add, edit, and remove layers from the canvas

1.5.1 Requirements to be tested

1. The user will be able to add and remove layers
2. The user will be able to modify the layer properties

1.5.2 Oracle

- Success
  - The layer that was selected and dragged to the slot will appear in the slot.
  - The status box displays the specific layer type has been added.
  - The slot that is selected and dragged to the trash icon is reset as an empty layer.
  - The status box displays the specific layer type has been removed.
  - Upon editing a slot’s properties, the system will not accept invalid data types and will update the correct slot’s layer attributes.
- Failure
  - Once dragged upon, the slot doesn’t update or updates to the wrong value.
  - Upon a slot being dragged to the trash icon, the slot doesn’t reset to empty.
  - An invalid property value is accepted and causes an error.
  - Slot properties aren’t updated upon the ok button being pushed when editing.

1.5.3 Expected Test Result

All layer types can be input into the empty slots, and their individual attributes will be those which correspond to their layer type. The drag and drop functionality will place the correct selected layer into the correct selected slot. All improper data updates will cause an error message to be displayed. All updates to layer properties will be saved and immediately updated via the gui. The correct selected layer will be set to empty and its displayed properties will disappear if dragged onto the trash icon.

1.5.4 Test Execution

STEP #	STEP DESCRIPTION	(INPUT) DATA	RESULT
1	Add one of each layer to canvas	Button click, drag and release on slot	The slot inherits the specified layer's attributes and displays them on the GUI.
2	Edit each slot's properties	Button clicks, text input	Invalid inputs cause a warning label to appear before saving. Valid inputs are saved to the slots properties
3	Delete each slot	Button click, drag and release on trash icon	Each slot is set to empty and its properties are no longer displayed.

1.5.5 Result Summary

Test Successful. All layers added were of the correct layer type. When the slots were individually clicked to edit, a popup with that slot's layer information was displayed with the current layer information. When cancel was clicked after editing, no changes were made. And, when the ok button was clicked, the value was updated in the gui. After editing the values, the layers were individually dragged to the trash icon and, when released, the slot was set to display an empty layer.

1.6 Memory Accessibility Test

User interface test - Ensure that the user can save canvases to their system for future user and load a saved canvas.

1.6.1 Requirements to be tested

1. The user will be able to save a canvas
2. The user will be able to open a previously saved canvas

1.6.2 Oracle

- Success

    - The canvas will be saved as canvas_name.pkl in the project path selected by the user
    - The canvas will be reopened with the same properties as when it was saved.
    - The canvas will be properly formatted when opened.
    - Invalid file types will not be opened
- Failure
    - The save file is not created, is in the wrong location, or has the wrong name
    - The opened file is corrupted or the canvas properties are different than what was saved.

1.6.3 Expected Test Result

When saved, a pkl file with the user input canvas_name will be saved to the user selected project directory. Upon opening a saved canvas, the current canvas will be overwritten with all the properties of the saved file. The opened canvas will be properly formatted and all saved properties are present.

1.6.4 Test Execution

STEP #	STEP DESCRIPTION	(INPUT) DATA	RESULT
1	Create a unique canvas with 4 slots.	Canvas property and layer information	The canvas is populated with no default values
2	Save canvas	Button push, canvas properties	The file is saved to the specified folder with the specified name
3	Open new canvas	Button push	All elements are cleared and set to the default view
4	Open invalid canvas	Invalid data path	The ok button could not be pushed/could not be opened
5	Load saved canvas	Input file path	All saved properties are present in the canvas

1.6.5 Result Summary

Test partially successful. The following applies to testing using a Linux OS or Mac OS 10.13.4. After creating a canvas with all non-default properties, the canvas was saved via a button click. When that was done the pkl file appeared in the correct folder. After saving the new canvas button was clicked and all canvas properties were set to default. Next the open button was pushed and the saved file was selected from the file browser. When ok was pushed all the saved values were set in the GUI.

This test cannot run using Mac OS 10.13.6. The menu bar in the system tray is not accessible by the user. Clicking "File" or "Help" does not create a dropdown menu.

1.7 File System Specification Test

Application test - Test that file and directory paths can be modified

1.7.1 Requirements to be tested

1. The user will be able to input a training data path
2. The user will be able to specify a model location

1.7.2 Oracle

- Success
    - A training data file path will be marked by the program to be used in the neural network training.
    - A folder will be marked by the program to save the training script and the trained model.
- Failure
    - The user will not be able to select a file or directory.
    - The program will not remember the location of the file or directory.
    - The program crashes.

1.7.3 Expected Test Result

By navigating to the canvas properties, the user will be able to click "browse" for the training data path or the project directory. This will open a file browsing tool native to the user's operating system. When selecting the training data path, the browser will not allow the user to confirm a selection unless they select a CSV file. When selecting the project directory, the browser will allow the user to confirm any directory they navigate to. After confirming a selection, the file path or directory will be displayed on the canvas properties box, and will be used when generating and training a model.

1.7.4 Test Execution

STEP #	STEP DESCRIPTION	(INPUT) DATA	RESULT
1	Open canvas properties	Mouse click	Pop up window opens containing canvas property options
2	Browse folders for training data	Mouse click on button	File browser window is opened
3	Select a CSV file to use as training data	File selection	File path is stored
4	Browse for the project directory	Mouse click of button	File browser window is opened
5	Select a folder to use as the project directory	Navigation of file browser	Directory path is stored

1.7.5 Result Summary

Test Successful. The canvas properties were opened and the file path and directory were set to the user's specifications.

1.8 System Status Test

User interface test - Ensure that the status box displays a status for user actions that alter the canvas design or interact with the backend.

1.8.1 Requirements to be tested

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

1.8.2 Oracle

- Success
    - The correct status for the given action is displayed by the status box.

    - An error status is output for neural network generation and model training that are passed invalid canvas designs.
- Failure
    - The incorrect status is displayed for an action.
    - No status is displayed for an action.
    - Errors in neural network generation or model training are not displayed by the status box.

1.8.3 Expected Test Result

The status box will update with a new status for user actions that alter the canvas design or interact with the backend. These actions include add/remove layers, opening a new canvas, generating the network script, training the model, cancelling the training of a model, and clearing all canvas slots of layers. Backend processes should print an error to the status box when the user attempts to generate or train an invalid canvas design .

1.8.4 Test Execution

STEP #	**STEP DESCRIPTION**	**(INPUT) DATA**	**RESULT**
1	Open the application	Python3 command in the terminal	The application opens and a welcome status is displayed in the status box
2	Add layers to create a valid neural network.	Mouse click and drag	Add Layer status displayed in the status box
3	Set the training data path in the Canvas Properties Box.	Filedialog input	Canvas points to valid training data CSV
4	Generate the neural network script.	“Generate Script” button click	The neural network script is saved to the directory. The generate script status is output to the status box.
5	Train the model	“Train Model” button click	Model training begins and the status is displayed to the status box.

6	Cancel training	“Cancel” button click	The model training is cancelled and the cancellation status is displayed by the status box.
7	Train the model to completion	“Train Model” button click	The model is trained and a model file is generated. The begin training, intermediate steps, and finished training statuses are displayed by the status box.
8	Alter Output Layer size to make design invalid.	Click on Output Layer. Text entry for size.	The Output Layer size is changed to 1 to make the canvas design invalid.
9	Generate the invalid neural network.	“Generate Script” button click	The neural network script is saved to the directory. The generate script status is output to the status box.
10	Attempt to train the model.	“Train Model” button click	The training start status is displayed by the status box. Then, a Keras training error is displayed by the status box, signifying unsuccessful training.
11	Remove a layer from the canvas.	Mouse click and drag	Remove Layer status displayed in the status box
12	Attempt to generate neural network.	“Generate Script” button click	A generation error status is displayed by the status box.
13	Clear Slots	“Clear Slots” button click	The slots are cleared of layers and the status box is updated with the clear status.

14	Create new canvas	“New Canvas” button click	The default canvas is displayed. The status box is cleared of all contents.

1.8.5 Result Summary

The test was successful. Each user action on the canvas design resulted in the status box displaying the correct status. Each user interaction with the backend resulted in the status box displaying the correct status. Trying to generate a neural network with empty layers resulted in a generation error status. Trying to train a model with a layer of an invalid size resulted in the status box displaying a Keras error.

1.9 Network Generation Test

Backend control test - Test the application’s ability to convert a user’s neural network design into code.

1.9.1 Requirements to be tested

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

1.9.2 Oracle

- Success
    - If there are no errors in the user’s neural network
        - The Python script is created and placed in the user specified project directory
        - A copy of the script is placed in the project repository
        - The script represents the neural network the user designed
    - If the user has entered an invalid network design
        - The status box displays any errors in the neural network logic
        - The Python script will not be generated
- Failure
    - The Python script is not created
    - The script is not copied into the project repository
    - Parts of the network are missing from the script or are wrong
    - The script is created despite errors

1.9.3 Expected Test Result

The user can drag and drop layer types onto the canvas to build a network. The user can then click the “Generate Network” button to create the Python script. If the user has entered an invalid network design, the status box will be updated to show the errors. Otherwise, the Python script will be generated and placed in the user’s specified project directory. The script will then be copied to the project repository to be imported by the system. The status box will be updated to indicate the script has been generated successfully.

1.9.4 Test Execution

STEP #	STEP DESCRIPTION	(INPUT) DATA	RESULT
1	Drag an input layer, a hidden layer, and an output layer onto the canvas	Mouse click, drag and drop	Layer properties are stored
2	Click the “Generate Network” button	Mouse click, canvas properties, layer properties	Canvas and layer properties are read in and the script generation begins
3	Status box displays “Network generated”	None	Python script has been created
4	Remove the input layer	Mouse click, drag and drop	Invalid network design created
5	Click the “Generate Network” button	Mouse click, canvas properties, layer properties	Canvas and layer properties are read in and checked
6	Status box displays “Generation error”	None	Errors detected correctly

1.9.5 Result Summary

Test Successful. A neural network was designed using three layers; an input layer, a hidden layer, and an output layer. A Python script was created in the specified directory which contained the three layers and the code needed to train and save a model. A copy of the script was also created in the project repository.

A second part of the test was conducted by removing the input layer, which is required for generation, and attempting to generate a new Python script. Instead, an error was displayed in the status box and nothing was generated.

1.10 Model Training Test

Backend control test - The the applications ability to read a CSV file and train a neural network model on the data.

1.10.1 Requirements to be tested

1. The user will be able to use the neural network to train a model
2. The user will be able to stop the current process without the crashing the application

1.10.2 Oracle

- Success
    - If a neural network Python script has been generated
        - A model is trained on the user's training data
        - The model is trained in a separate process
        - Training can be stopped at any time
        - Model files are saved to the user specified project directory
    - If a neural network has not been generated
        - The status box will display an error message telling the user to generate a network
- Failure
    - The model does not successfully train
    - The GUI is unusable during training
    - The training process cannot be cancelled
    - Some or all model files are missing from the project directory
    - The neural network function cannot be imported by the system

1.10.3 Expected Test Result

This test assumes a neural network Python script has already been generated. If not, the status box will be updated to display an error message telling the user to generate a neural network. The user will click the "Train Model" button to begin the training process. This will spawn a new process that will take care of training a model on the user's data, and occasionally pipe back status messages. The user will cancel the training process without the program crashing using the "Cancel" button, and start a new training process. Once this process finishes, the model files will be stored in the project directory.

1.10.4 Test Execution

STEP #	STEP DESCRIPTION	(INPUT) DATA	RESULT
1	Generate a neural network script (1.9)	Mouse click, canvas properties, layer properties	A Python script representing the neural network is generated
2	Click the "Train Model" button	Mouse click	A model begins training
3	Click the "Cancel" button	Mouse click	The training process is cancelled
4	Click the "Train Model" button again	Mouse click	A new model begins training
5	Status box displays model accuracy	None	The model has been trained

1.10.5 Result Summary

Test Successful. A neural network was designed using three layers; an input layer, a hidden layer, and an output layer. A Python script was then generated from this design. Once a script was generated, the "Train Model" button was pressed and the appropriate messages were displayed in the status box. After waiting approximately 10 seconds, the "Cancel" button was pressed and the status box was updated. No errors were encountered and the application did not crash. The “Train Model” button was then pressed again and training was allowed to finish. Training finished with no errors and the model accuracy was displayed in the status box.