Software Engineering Final Project

Interim Project Report

Colin Grandjean, Lauren Giles, Cole Johnson, and John Runyon

New Mexico Tech Institute of Mining and Technology

CSE 326: Software Engineering

Dr. Dongwan Shin

1. Project Overview

OCR (Optical Character Recognition) is a technique that a program can use to parse individual or stream images into a matched set of some written alphabet; often just a set of alphanumeric characters. OCR is used in banking, note taking applications, and many other services used on a daily basis. One of the most common implementations of OCR is the use of neural networks, often through supervised learning methods. We propose to create a simple neural network that will be trained and tested to classify a single image into numeric characters using one such supervised learning technique. In the process of developing the neural network, we'll develop a host of tools to test and view the networks created and some performance metrics to gauge effectiveness.

There are a few motivations for the project. First, the development of this tool can help people learn more about neural networks and how they work using our graphical user interface. Second, OCR has a range of applications – such as written text to digital transcription – that could be adapted from the code base we aim to create.

1.1 Project Scope and Objectives

The initial scope of the project, as outlined in the overview, will be to create a simple Optical Character Recognition (OCR) system using a neural network and employ supervised learning techniques to test and train our model. Our main objectives for the project include:

1.) Build the components of a neural network using an object oriented approach (using encapsulating class like Neuron, Layer, and NeuralNetwork)
2.) Create a training and test environment for the neural network
3.) Provide a graphical user interface (GUI) that the user can draw characters for the model to classify
4.) Provide a graphical user interface (GUI) where the user can view the neural network

1.2 Supplementary Requirements

The non-functional requirements for the project involve four main components: Extensibility, Reusability, Reliability, and Documentation. Here is how each of these components are being addressed in our implementation and design.

1.) **Extensibility:** Using an object-oriented approach, data and behaviour of each substructure of the neural network is encapsulated and abstracted. For example, Layers are equipped with a generic activation and loss function, allowing the architecture to use combinations of specific implementations of each of these functions.

2.) **Reusability**: The project components of the project should be easily reusable. One way this goal is accomplished is through our class Serializer implementation. This allows us to save the configuration of a Neural Network after training is completed.
3.) **Reliability:** The project should have assurance measures to help determine and control the quality of code. To accomplish this goal, we have implemented unit tests to help ensure the functionality of each subsystem is correctly implemented.
4.) **Documentation:** To help manage the project, measures that document the development and management of the project need to be implemented. First, to document development, we use *GitHub*, a version control system that tracks individual contributions to the codebase. Second, to document management, we use *Jira* to track development tasks and assign them to members of the project.

2. Customer Requirements

Using a use-case model of our requirements, main prioritized use-cases identified were the following:

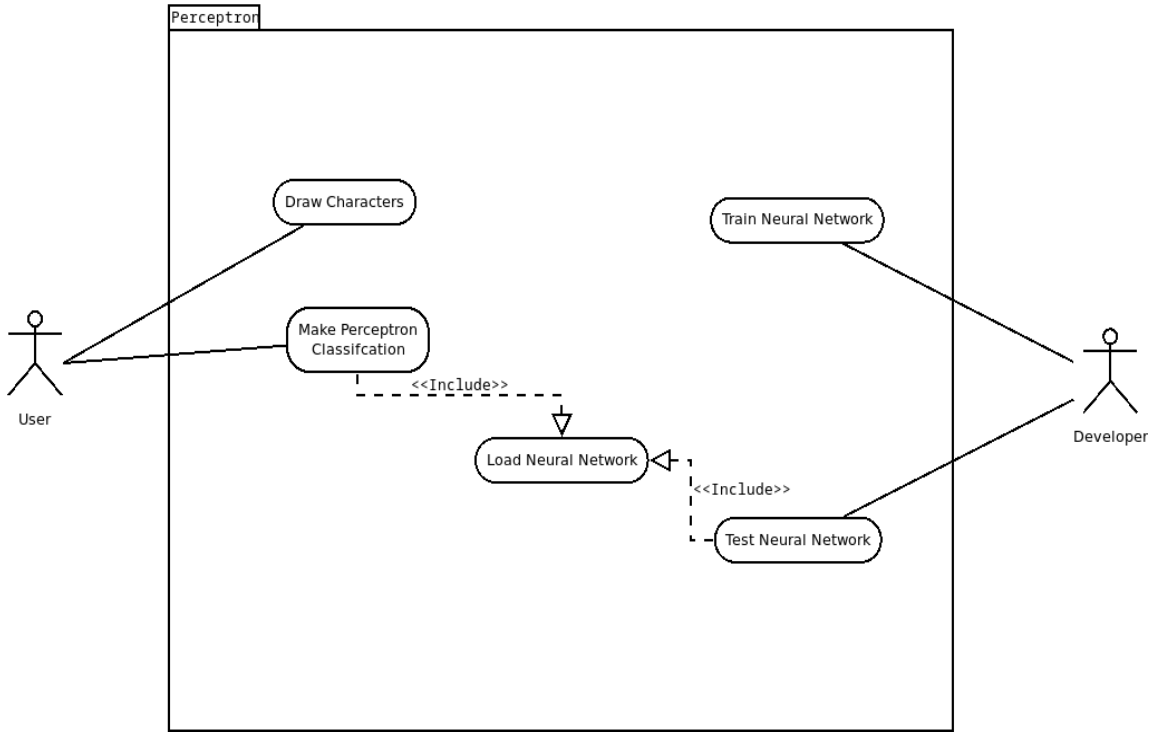


Figure 1.1: Prioritized Use-Cases

In our use-case diagram, we consider two main actors: the **User** and the **Developer**.

a. **User:** the user's main activities involve GUI interaction, mainly drawing characters, and making classifications using the neural network on those drawn characters.
b. **Developer:** the developer's main activities involve interaction with the base neural networks or perceptrons. The developer should be able to test and train a neural network given a starting configuration and a database.

The prioritized use-cases identified for the project were as follows:

a. **Draw Characters:** draw characters using the GUI.
b. **Make Perceptron Classification:** make the classification.
c. **Train Neural Network:** train network based on implemented LossFunciton and ActivationFunction passed on the layer level.
d. **Test Neural Network:** test network, mostly tests accuracy on labeled testing data.

3. Architectural Design

The architecture of the project is an adaptation of the N-Tier model, where the system is divided into hierarchical tiers of increasing abstraction–where each tier will encapsulate and abstract the behaviour of the previous tiers. We decided on this approach because neural networks can be more naturally divided into smaller substructures that are encapsulated by other structures. For example, the most fundamental part of a neural network is, of course, the neuron. Bigger substructures are built on top of the neuron. The *Layer* class, for instance, is built from a collection of neurons. So, the structure of the neural network meshes well with the bottom-up nature of the N-Tier model.

We considered a few other architectural design patterns, such as the Model-View-Controller (MVC) where the GUI would become a go between the Controller and View, where it would make requests to the controller and show the results produced by the view. However, this approach would collapse a lot of the important components–like the *Neuron*, *Layer*, and *Neural Network*–into a singular component, which would flatten a lot of differences we would like to highlight during design.

3.1 Subsystem Architecture

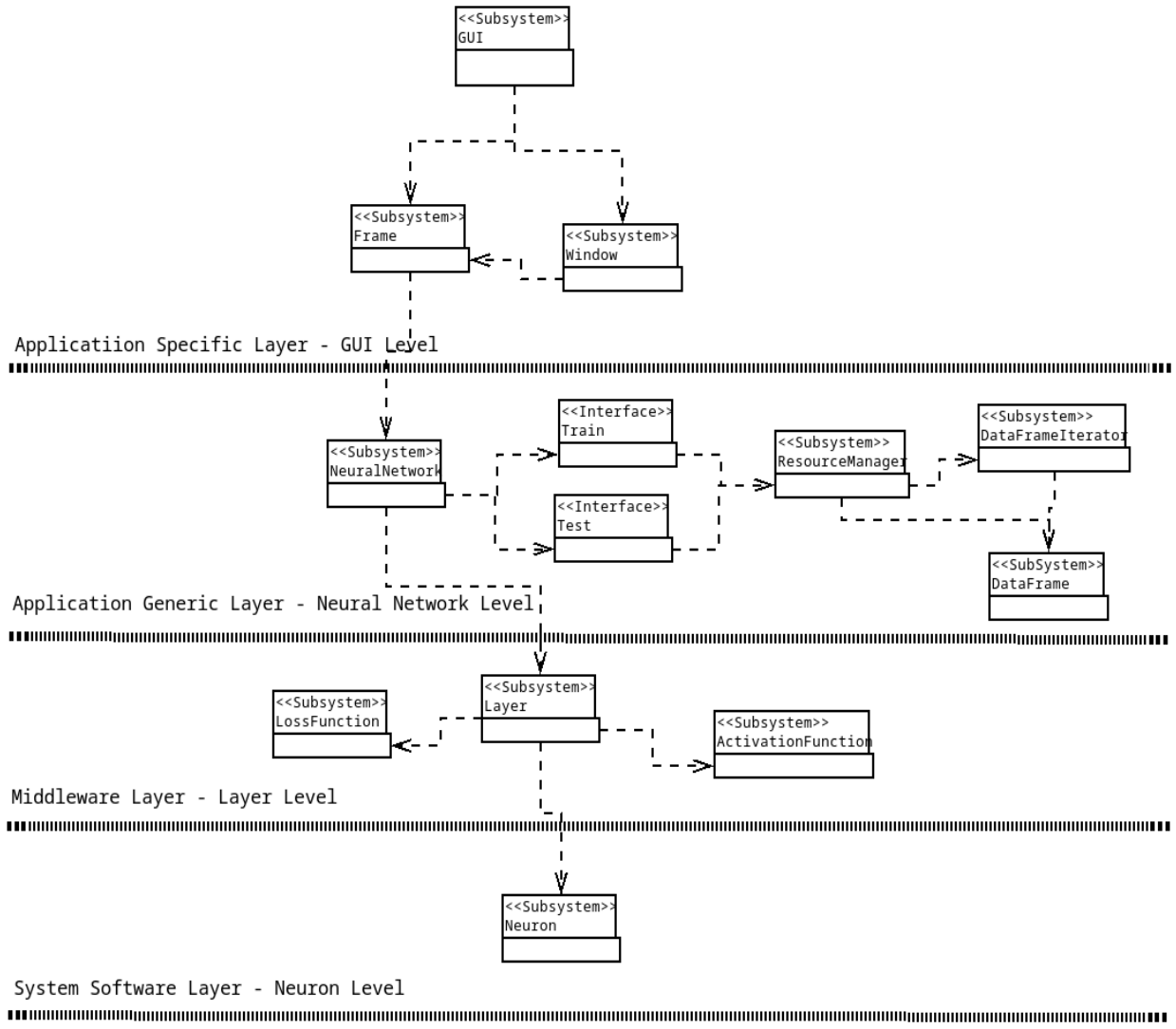


Figure 3.1: Subsystem view of the project

3.2 Deployment Model

The deployment of our application doesn't require many third party dependencies. For the most, the Java Runtime Environment executes the compiled byte code for the program and serves as the deployment for almost all of our applications.

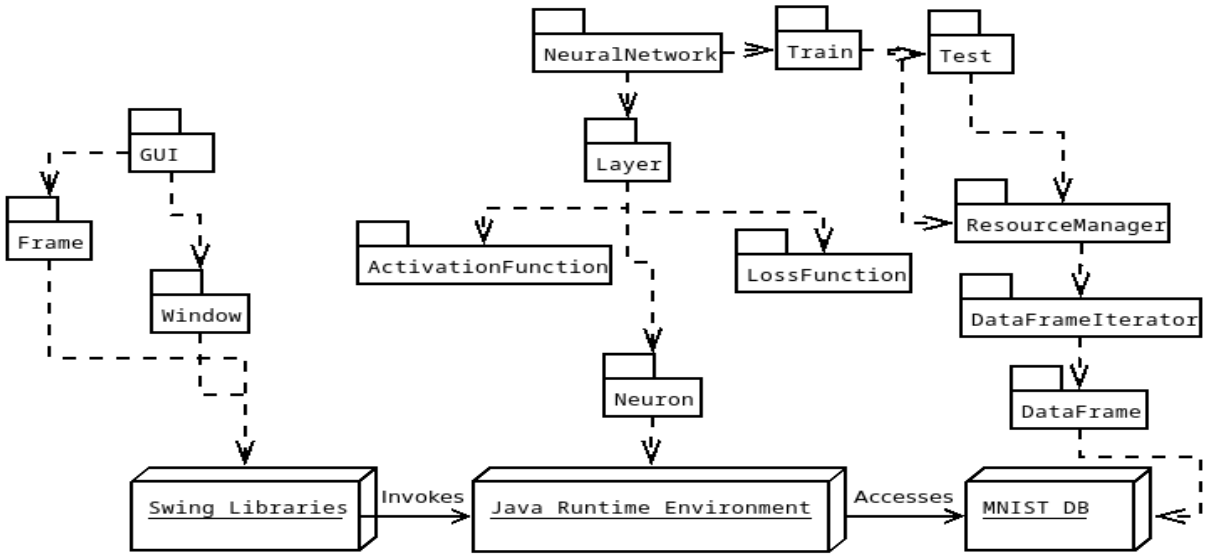


Figure 3.2: Deployment model diagram of the project

4. Use Case Realization Design

4.1. Make Classification Use Case

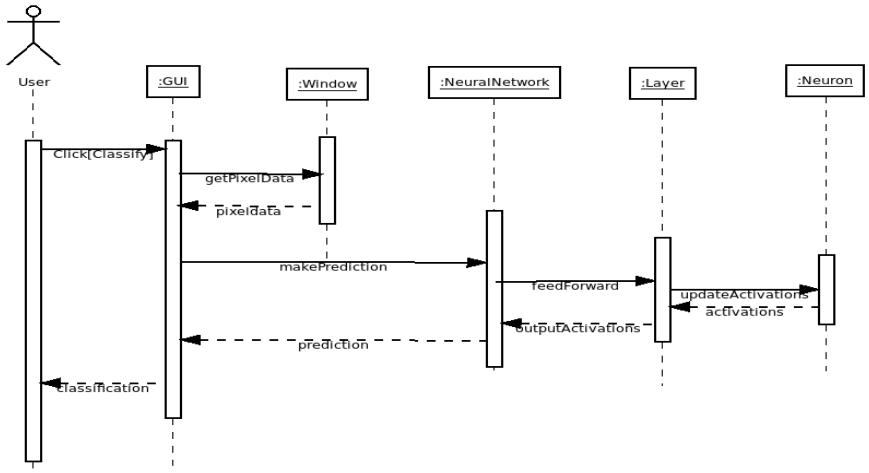


Figure 4.1.1: Use-case realization of user classification via sequence diagram

In order to realize the classification function, method layers carry down through all the layers of N-Tier architecture: *GUI, NeuralNetwork, Layer,* and *Neuron.* Here, our design method allows for cascading delegation to lower level functionality in more primitive, base layers of the system. The following class diagram show the use-case realization from a class perspective:

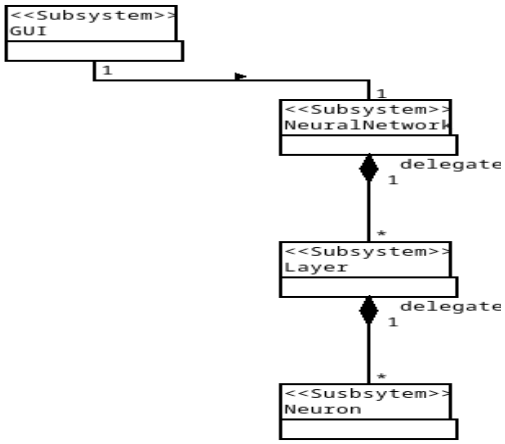


Figure 4.1.2: Class diagram of use-case realization

4.2 Draw Characters

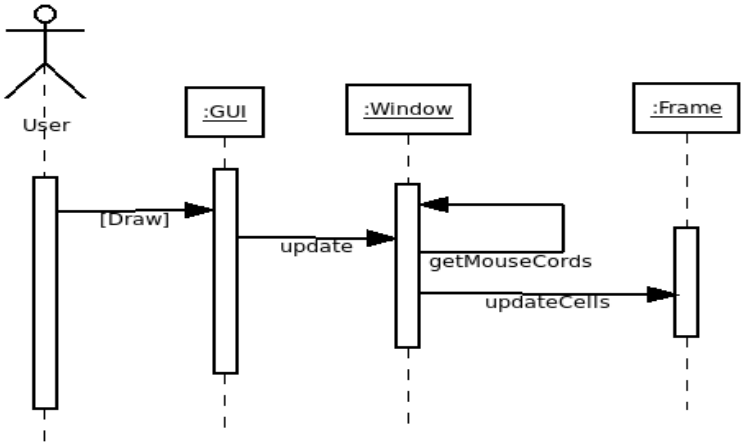
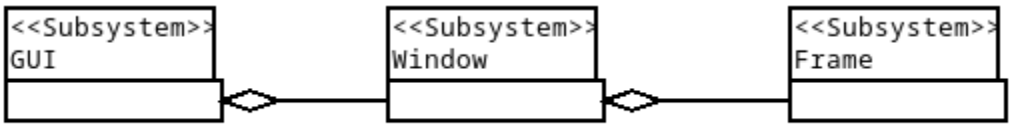


Figure 4.2.1: Sequence of use-case realization of drawing characters

In order to draw characters, most of the GUI components need to interact with each other. In this case, the GUI–via an ActionListener–calls a method to update the window, and then this task is further subdivided on the frame level, where mouse input is recorded and the underlying array data is changed.



5. Subsystem Design

5.1 Subsystems Overview

The OCR will be structured into a few components as listed below:

1. **GUI** – Provides user interaction, including drawing characters and visualizing the neural network through Swing in Java.
2. **Neural Network** – Implements the core neural network functionalities (training, testing, and classification).
3. **Testing & Training** – Ensures correctness and measures model accuracy. Done through unit testing and debugging the team. Manages file input/output, dataset loading, and precomputed trained model storage.

5.2 Subsystems Descriptions

5.2.1 GUI

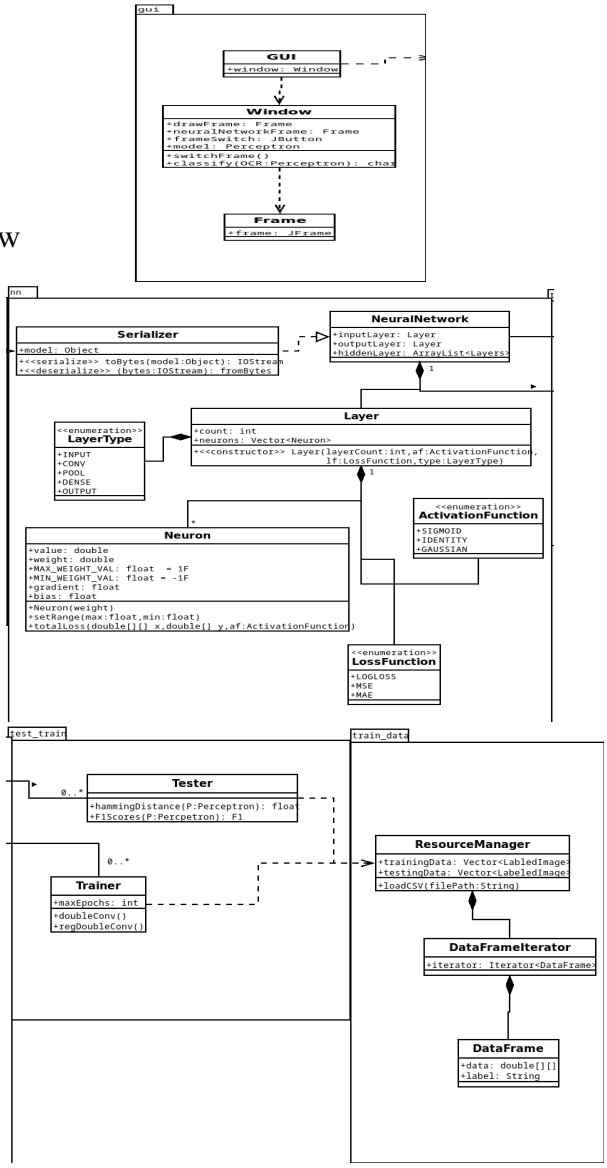
- Technology: Java Swing and AWT
- Responsibilities:
  - Provide tools for the user to draw
  - Display neural network
  - Show classification of character

5.2.2 Neural Network

- Technology: Java
- Responsibilities:
  - Train, test, and classify characters

5.2.4 Testing and Evaluation

- Technology: JUnit
- Responsibilities:
  - Validate individual components
  - Measure accuracy and performance



5.3 Interactions Between Subsystems Example

NeuralNetwork: Processes the input and makes a classification.

GUI: Displays the result, and takes the user input.

Testing & Training: Validates performance and functionality.

6. Human Interfaces

We currently have a working prototype of the interface that lets the user draw a character and submit it. This includes an area to draw something, a button that clears the drawing section, and a submit button, which saves the drawing to an image file. We are still deciding the formatting required from the user interface in order for the Neural Network to interpret it. The interface is not integrated with the neural network yet, but will be ready to send the image to the network. The following screenshot shows the interface and an example of a user's input.

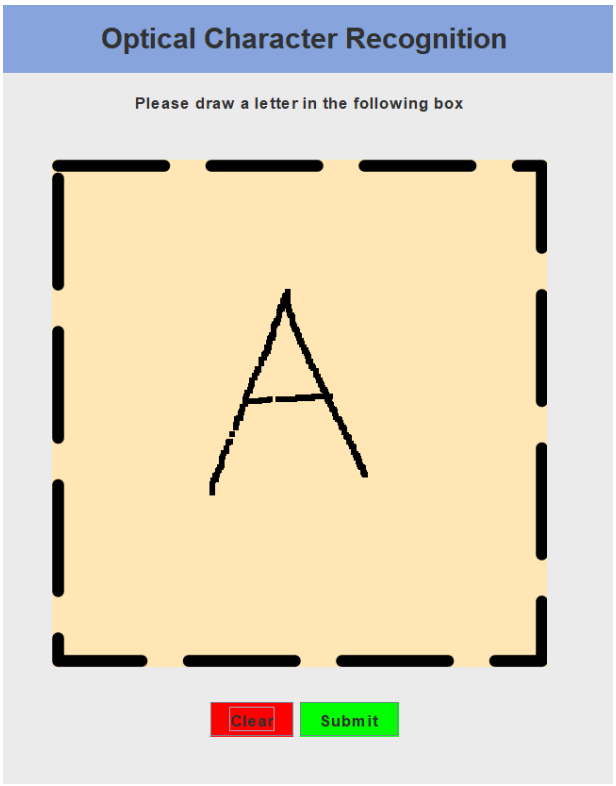


Figure 6.1: Example user input on the GUI

There is one more part we are still developing that will show a visualization of the neural network. A mockup of this visualization is shown in the following screenshot.

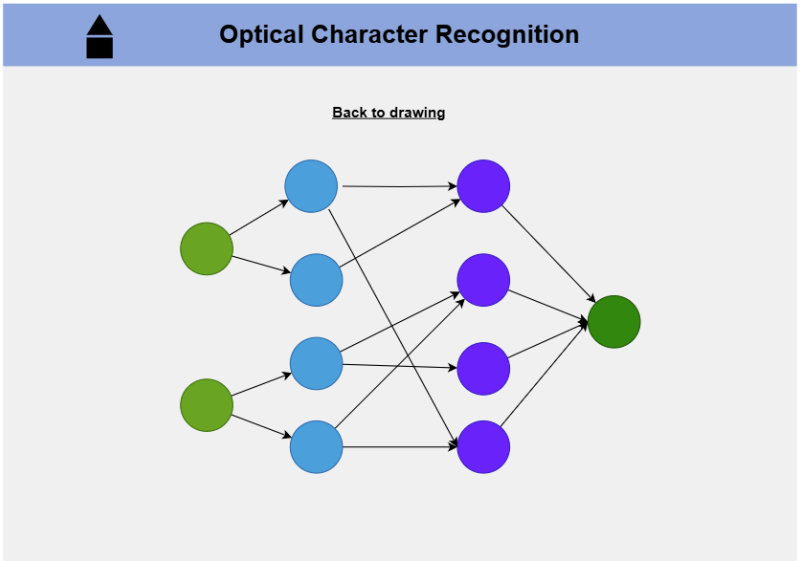


Figure 6.2: Mockup of the visualization of the neural network

7. Testing Plan

Our testing plan will include methods to ensure that our program is not just correct (as in it merely functions), but that it is also very efficient, reliable and not commonly susceptible to random, inconsistent bugs. This will be done through unit testing using JUnit to individually test each major component of our code throughout the development lifestyle. This will also include Neural Network validation and training to make sure classifications of characters are consistently correct.

8. Appendices

All members contributed equally to the project.

8.1 Project Status

Status on Components of the Project

GUI:

Simple GUI that allows the user to draw a character and have an image produced from that character has been created. We are currently working on having the GUI interact with the neural network in its full capacity. We are also working on having a visual representation of the NN after an image is submitted.

Neural Network:

Neural network is slightly behind schedule. We do have the test functionality up and running, but only partial completion of the training functionality. We will put a lot of focus into catching up in this area, as it is the most critical part of the system.

Data Handling:

We are currently working on having the data received from the user's drawing converted into either a CSV of coordinate points, or into a DataFrame. Additionally, we are in progress with storing training data and model parameters.

Testing and Evaluation:

As of this moment, we are able to do some simple testing of the neural network, however full scale testing as well as the ability to measure the networks accuracy and performance is in the works.

Overall Status of the Project

Overall, we believe that we have done much of what we were aiming to have accomplished by this stage in development. We have completed the planning and preparation phase, and have already made significant progress in the development of our GUI and neural network systems. We intend to begin fully testing our system in a month, so that we can work towards our goal to continue to build upon and refine what we have already been able to achieve, and have a fully functional optical character recognition program by the project presentation date.