Contents

[Introduction 2](#_Toc58006620)

[Customers 2](#_Toc58006621)

[Team 6](#_Toc58006622)

[Design 7](#_Toc58006623)

[Design Process 7](#_Toc58006624)

[System Design 7](#_Toc58006625)

[Design Diagrams 8](#_Toc58006626)

[Use Case Diagrams 8](#_Toc58006627)

[Class Diagrams 11](#_Toc58006628)

[Sequence Diagram 13](#_Toc58006629)

[Rationale Management 16](#_Toc58006630)

[Verification 17](#_Toc58006631)

[Preprocessing GUI: 18](#_Toc58006632)

[Model GUI: 20](#_Toc58006633)

[Testing GUI: 22](#_Toc58006634)

[Reflection 24](#_Toc58006635)

[Bibliography 26](#_Toc58006636)

QuickML

|  |  |
| --- | --- |
| **Juan Lopez** | **Thony Yan** |

# Introduction

Quick Machine Learning (QuickML) is a computer application to introduce users to Machine Learning. Through this application users will have the power to use Machine Learning, specifically computer vision, with no need of knowing how to program or the complexity of Machine learning.

## Customers

1. We created this application for the intent of educational purposes, so anyone who wishes to explore the applications of computer vision in Machine Learning may use it.
2. Our team will design a user-friendly and straightforward software where anyone with basic computer knowledge will implement their own Machine Learning algorithm.
   1. Machine Learning and Artificial Intelligence have gained prominence in recent years, according to Indeed, Machine Learning Engineers have a 344% growth in job posting in the year between 2015 to 2018, making it the rank 1 growth in rate. Another Machine Learning related job also ranks at number 13, Computer Vision Engineer (Daws, 2019). with companies like Google, Microsoft, and Amazon creating a cloud-based platform for Machine Learning. There are varieties of ways to use Machine Learning from image and voice recognition, solving complex problems, detecting anomalies, self-driving cars, autonomous systems, etc. Clearly, Machine Learning is a very important field that will help shape a better future. But, as it is a relatively new and exploding field with many fields put together like Calculus, Statistics, and Programming involve, everyday people will shy away from using Machine Learning because of its complexities. Therefore, we need to introduce individuals to Machine Learning as there’s a severe need for Machine Learning talent (Terra, 2020). Our product, QuickML, will help ease this gap by creating a software that will allow individuals to have the power of Machine Learning without needing to know the complex fields in it.
   2. Although there are other tools and applications in the market, we have found none straightforward ones. They made these applications with more complexity as they were made for users with more experience and knowledge in the field. Google makes another online tool called Teachable Machine. This is an extremely simple Machine Learning online application tool, where all you must do is input the images you want to classify. It works very well but it would not be an application to use when trying to learn Machine Learning as it has very limited configurations.
   3. QuickML would be a playground where you man changes parameters to make a model, train a model, and see and test the results. In this way users can start getting an interest in Machine Learning and will pursue more knowledge on it.
3. Contextual issues and external constraint:
   1. As we are making a Graphical User Interface (GUI) for QuickML through the Python programming languages in Windows 10 operating system, we do not how it will perform in another operating system like Linux or Mac. We will use PyQt to create the GUIs and from the documentation. It says that it should be compatible with that other operating system, but we do not know to what extent.
   2. Also, we will most likely put some limitations to the parameters that users can input into the software in the machine learning model creation process, as having a very large number will cause instability when training the machine learning algorithm. Special hardware known as Graphical Processing Unit (GPU) needs to be accessible for bigger models.
4. Schedule outline:

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **QuickML**  As a User I want to make and test a Machine Learning model | | | | |
| Epic | User Story | Task | Story point | Done? |
| 1. Preprocessing GUI | 1.1) As a user, I want to organize data to get it ready for the training step. | Create folder directory structures. | 5 | Checkmark |
| Copy from one directory to corresponding directory | 8 | Checkmark |
| 1.2) As a user, I want to see the labels to know I put the right label name. | Let the user input the labels they want. | 1 | Checkmark |
| Show the labels the user has inputted | 2 | Checkmark |
| Let the user delete inputs | 3 | Checkmark |
| 2. Model GUI | 2.1) As a user, I want to make different configuration to see the effects of the model. | Clearly put what type of layer we can put into model | 1 | Checkmark |
| Allow the user to put inputs for configuration. | 3 | Checkmark |
| Show the model. | 2 | Checkmark |
| 2.2) As a user, I want to see the progress of the training so I can know how the process is going. | Put a progress bar. | 5 | Checkmark |
| 3. Model Testing GUI | 3.1) As a user, I want to load my model to test if the training worked. | Create a prompt for the user to pick a file. | 1 | Checkmark |
| 3.2) As a user, I want to pass an image in the GUI and get a result. | Create a prompt for the user to pick a file. | 1 | Checkmark |
| Show result | 5 | Checkmark |

|  |  |  |  |
| --- | --- | --- | --- |
| 1. Preprocessing GUI | 1.1) As a user, I want to organize data to get it ready for the training step. | Verify that the user can just pick the folder with the images and give it a label name and is structured correctly for training step. | Checkmark |
| 1.2) As a user, I want to see the labels to know I put the right label name. | Verify that the user can see the labels on a text box and have the functionality to delete a label and its corresponding directory deleted. | Checkmark |
| 2. Model GUI | 2.1) As a user, I want to make different configuration to see the effects of the model. | Verify that the user can make different configurations to make a Machine learning model | Checkmark |
| 2.2) As a user, I want to see the progress of the training so I can know how the process is going. | Verify that the progress bar in the GUI slowly increases as training is happening. | Checkmark |
| 3. Model Testing GUI | 3.1) As a user, I want to load my model to test if the training worked. | Verify that the user can put a model file and be able to load the model to the GUI. | Checkmark |
| 3.2) As a user, I want to pass an image in the GUI and get a result. | Verify that the user has a way to input or drop an image to predict the image. | Checkmark |

* 1. For the user story allocation, we use Story Points and order the importance using the Fibonacci sequence. Each GUI has their own Story Point sequence.
  2. The Sprint period is one week consisting of 4 sprints.
  3. Sprints: *User Story (U.S.)*

1. **Sprint 1: Preprocessing GUI** *Epic 1*

* Create GUI layout *(U.S. 1.1)*
* Have a text input for user label inputs and directory paths. *(U.S. 1.1)*
* Make a text windows that show folder paths and label names *(U.S. 1.1)*
* Create a way to delete Labels and corresponding folder path *(U.S. 1.1)*
* Use inputs to create necessary directories to move images into *(U.S. 1.2)*
* Move images from source to destination path *(U.S. 1.2)*

1. **Sprint 2: Preprocessing GUI / Modeling GUI** *Epic 1 / Epic 2*

* Fix any problems from preprocessing that might cause instability to the next GUI. *(Epic 1)*
* Create GUI layout. *(U.S. 2.1)*
* Have a text window that shows the current model. *(U.S. 2.1)*
* Have an easy layout to put model structure building blocks. *(U.S. 2.1)*
* Once everything is done, set some constraint to avoid creating unstable or impossible models. *(U.S. 2.1)*
* When finish creating the configurations, run the training process and create a progress bar to see progress. *(U.S. 2.2)*

1. **Sprint 3: Model Testing GUI** *Epic 2 / Epic 3*

* Fix any problems from modeling GUI that might cause instability to the next GUI. (*Epic 2)*
* Create GUI layout. *(U.S. 3.1)*
* Create a way to load a model. *(U.S. 3.1)*
* Have a function to put input image. *(U.S. 3.2)*
* Have one image box showing the image. *(U.S. 3.2)*
* Have a text box showing the classification. *(U.S. 3.2)*

1. **Sprint 4: Wrap up** *Epic 1 / Epic 2 / Epic 3*

* Solve any necessary problems that can cause a software crash.
* See if we can improve any previous GUI to be more user friendly.
* Verifications

## Team

1. Thony Yan–Programmer / Machine Learning Engineer
   1. Motivation
      1. My motivation is simply trying to give back to the community, my university Florida International University has always advertised the concept of lifelong learning and giving back to the community. Seeing as Machine Learning has exponentially become a popular subject. I feel like the only way we can advance in this field is to have a bigger community interested in this field to have more collective knowledge and improve Machine Learning for a brighter future. To do this, we must first get the community to be interested in this subject, so I believe letting people play around and see the results would ease their way in.
      2. This will be useful in my career as I plan to work as a Machine Learning Engineer. This would that I can be a candidate to be a Software Engineer or Machine Learning Engineer by making applications and gaining experience in the Software process.
   2. Role
      1. Programmer and Machine Learning engineer will handle most functionality that deals with making the Machine Learning aspect functional
      2. Have about two years’ experience with Machine Learning mostly on Computer Vision
2. Juan Lopez–Programmer / GUI Designer
   1. Motivation
      1. This is a pretty wonderful project to use because neural networking is very important. I may not have any experience right now with neural networking, but I have a lot to gain from it.
      2. Neural networking recognizes the underlying relationship in data to solve issues. For example, in the company I work with, we often must process IDs and ensure they are not fake. If we had neural networks that processed them, it could speed up the processing of ID’s and eliminate any ID that is not valid or potentially fake. With a car, we would want it to identify what is a human and what isn’t a human so it can avoid any fatalities. We can use neural networks in so many ways, so learning about them can be relevant for almost any career.
   2. Role
      1. The designer of the GUI makes the program look as pretty as possible. I must ensure the GUI also functions properly with the code in the background so that the neural network works properly.

# Design

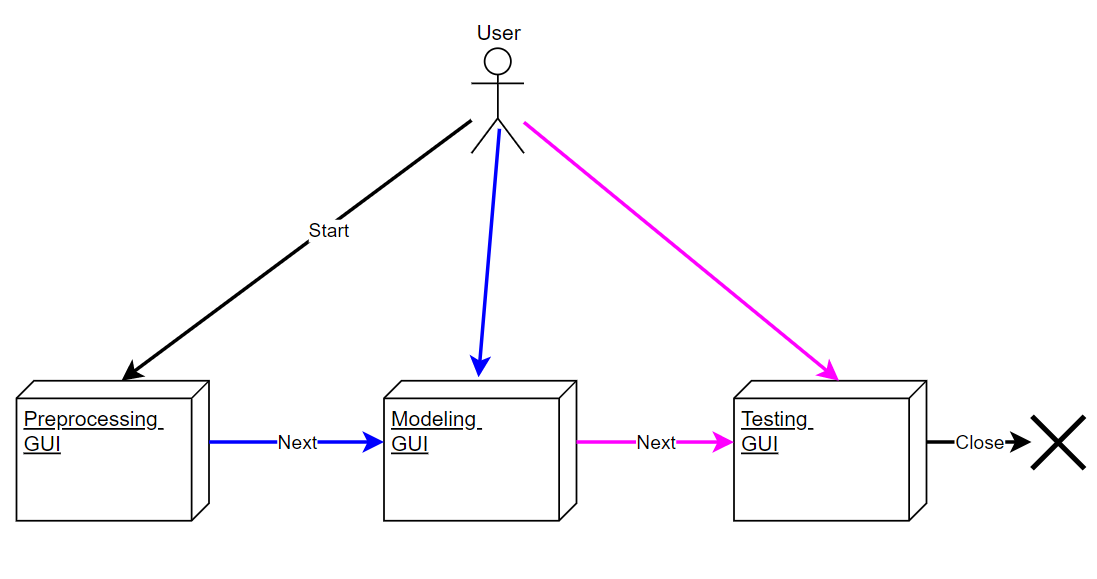
Software design is the process of defining the software’s architecture, components, interfaces, and other characteristics of a system or component. Also, software design is the software engineering life cycle activity in which we analyze software requirements in order to produce a description of the software’s internal structure that will serve as the basis for its construction. QuickML software design will follow the Component-Based Design (CBD) strategy, this approach is creating well-defined interfaces and dependencies that can be composed and deployed independently (Pierre Bourque, 2014).

## Design Process

The way the design process went for QuickML was by looking at other similar applications first to see what they offer. From the information gathering, we found one application that would fit the criteria, but we believed it was too advanced and not user-friendly for beginners. As a team, we first needed to share how the process should go. When doing computer vision with Machine Learning the process is usually Preprocess data, make model, train model, and test model. Through this order, we made three GUIs, the first one to preprocess the data, the second to make and train the model, and the last one to test the model. With that process in mind, we move on to create use case diagrams, sequence diagrams, and class diagrams.

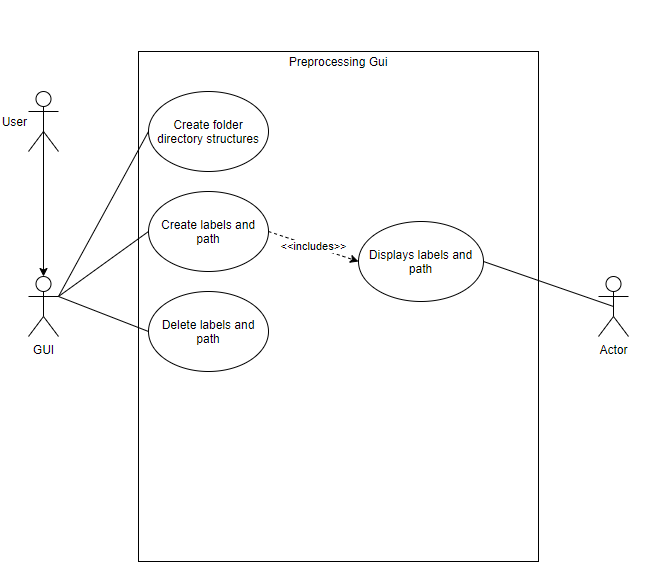
## System Design

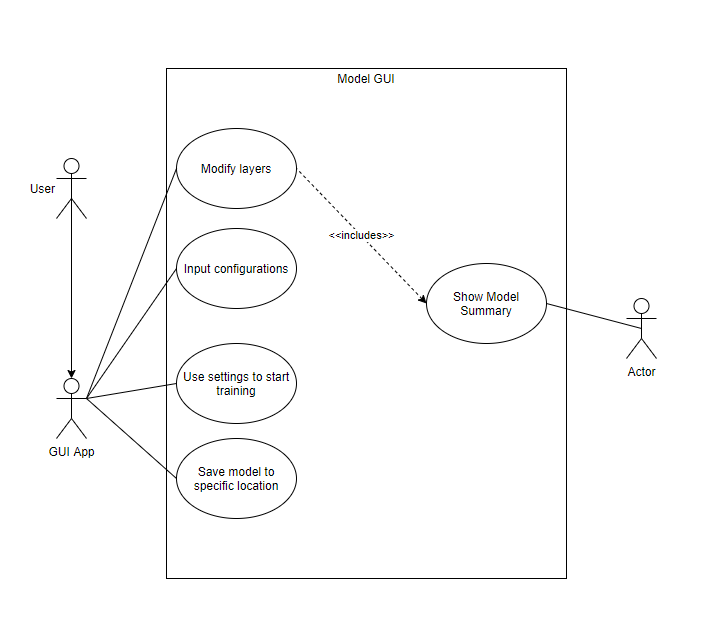
Here we have the system design, the user will start with a view of the Preprocessing GUI. When the next button is clicked then the program will close the Preprocessing GUI and show the Modeling GUI. The same functionality will be implemented when going from the Modeling GUI to the Testing GUI.

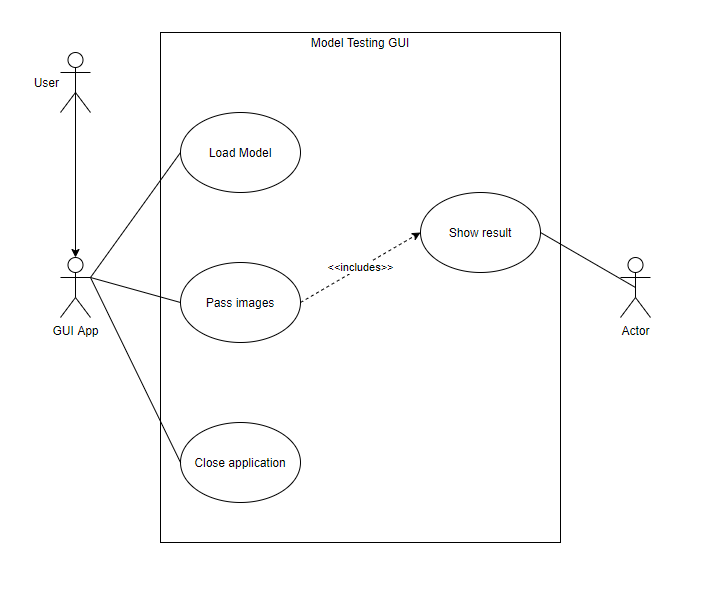


## Design Diagrams

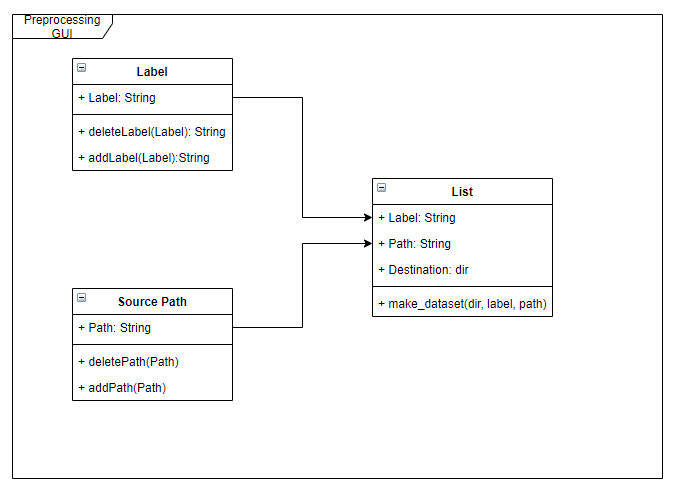
### Use Case Diagrams

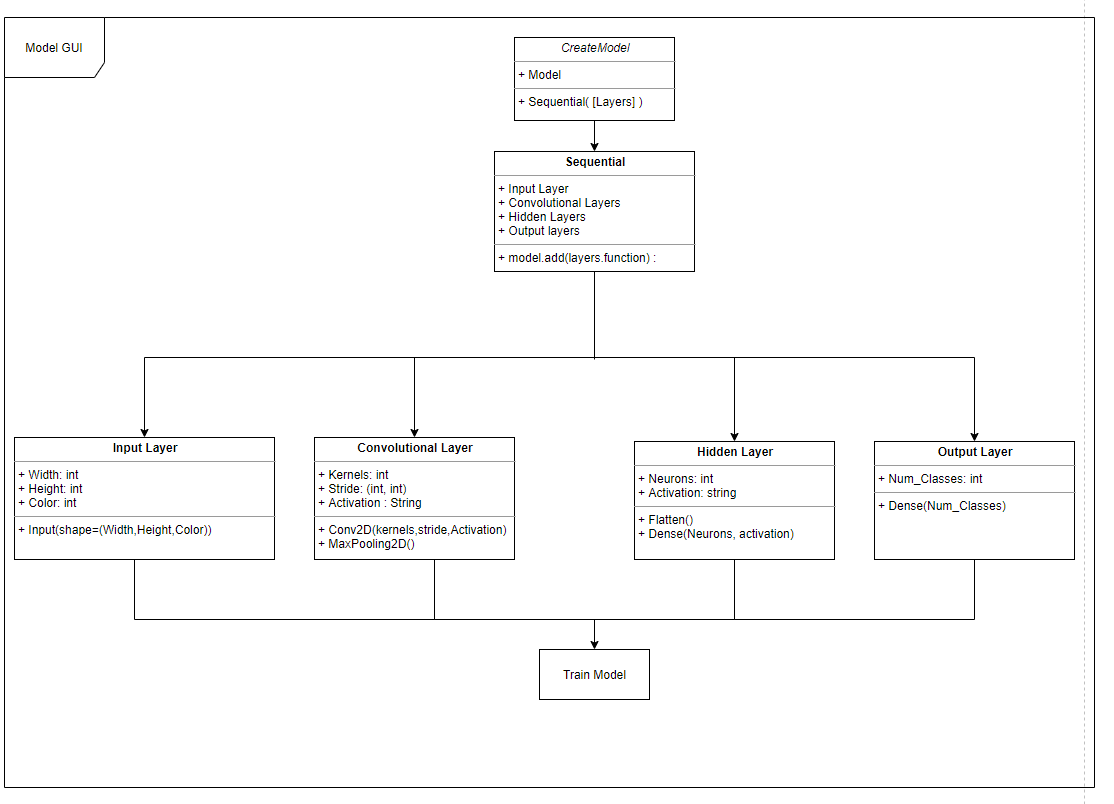


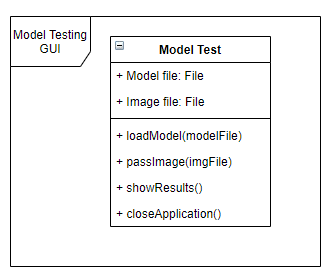




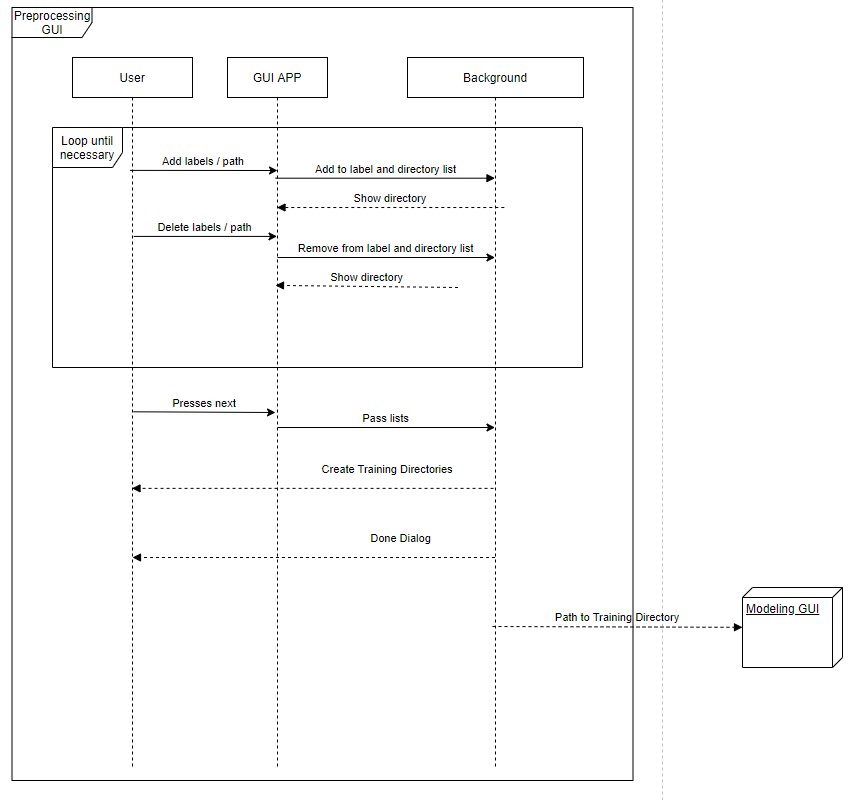
### Class Diagrams

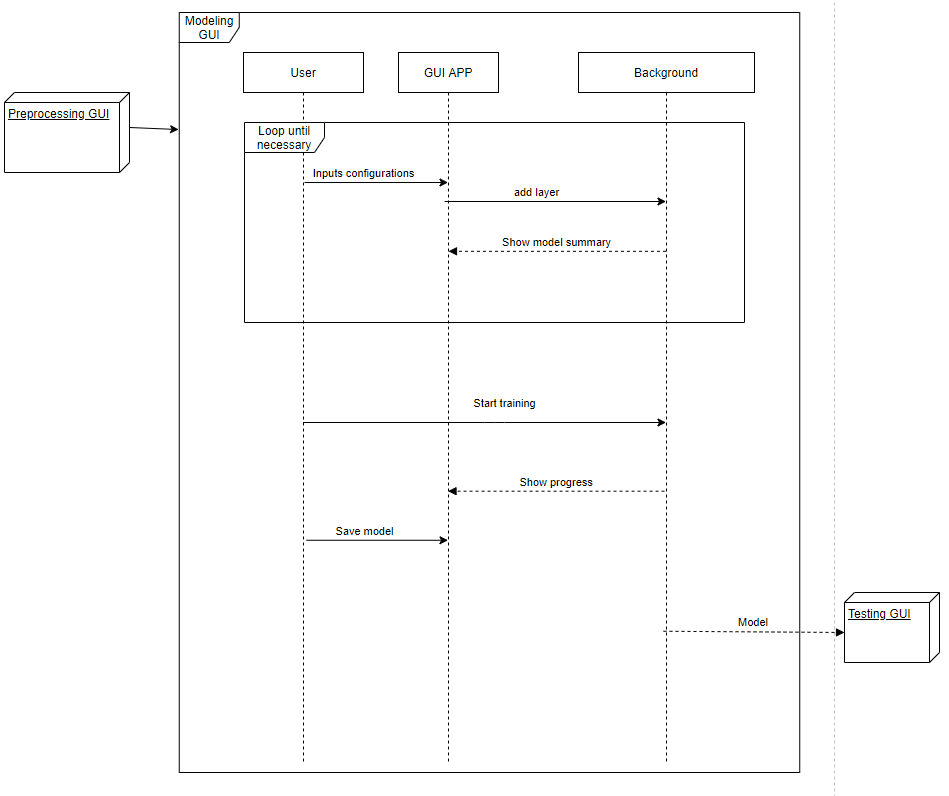


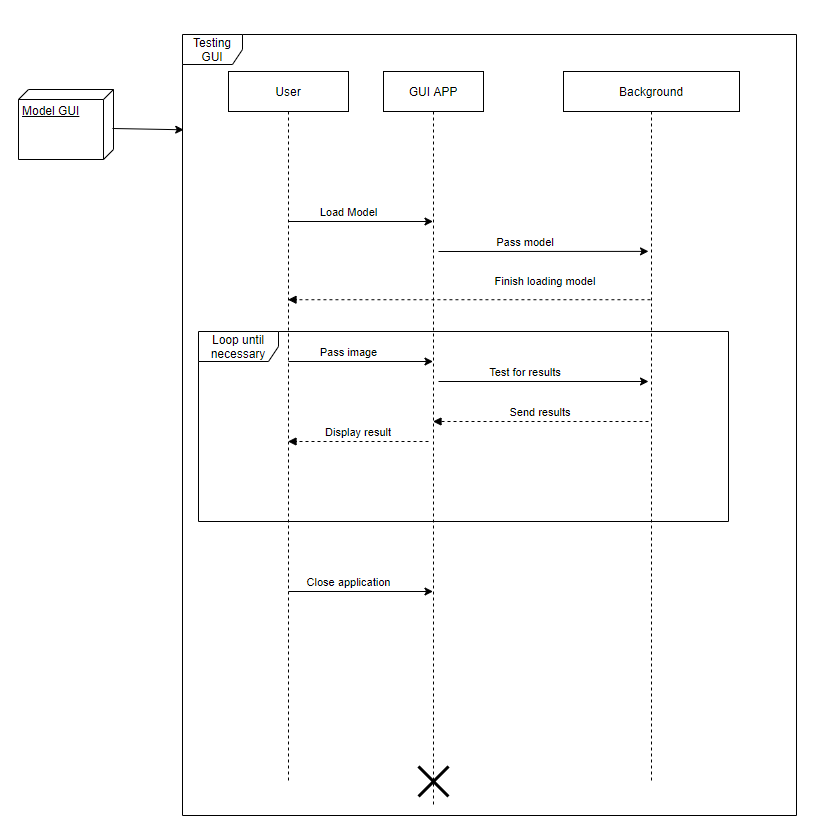




### Sequence Diagram







## Rationale Management

There were many design decisions we had to make throughout the project. However, we stuck true to the ultimate goal of the project. The criteria that guided every decision was to make a neural network that any user can test and understand getting them interested in working with neural networks. The first issue was simplifying neural networks in a way that people of many different technical backgrounds can understand, but we also had to keep in mind that most of the users would already have some knowledge on nerve networks. Therefore, we had divided it into three GUIs, the preprocessing GUI, the modeling GUI, and the testing GUI. Then we simply had to make it so the user would understand what they would have to do with the application. A lot of the input would come in from the head programmer, and then the GUI designer would give feedback. We came to this decision because the head programmer already had more experience with neural networks, while the GUI designer had very little to no experience with neural networks. This leads to very little debate as it fulfills the common goal of allowing users with little experience to experiment with the world of neural networks, yet also allowing more experienced users to also use the program.

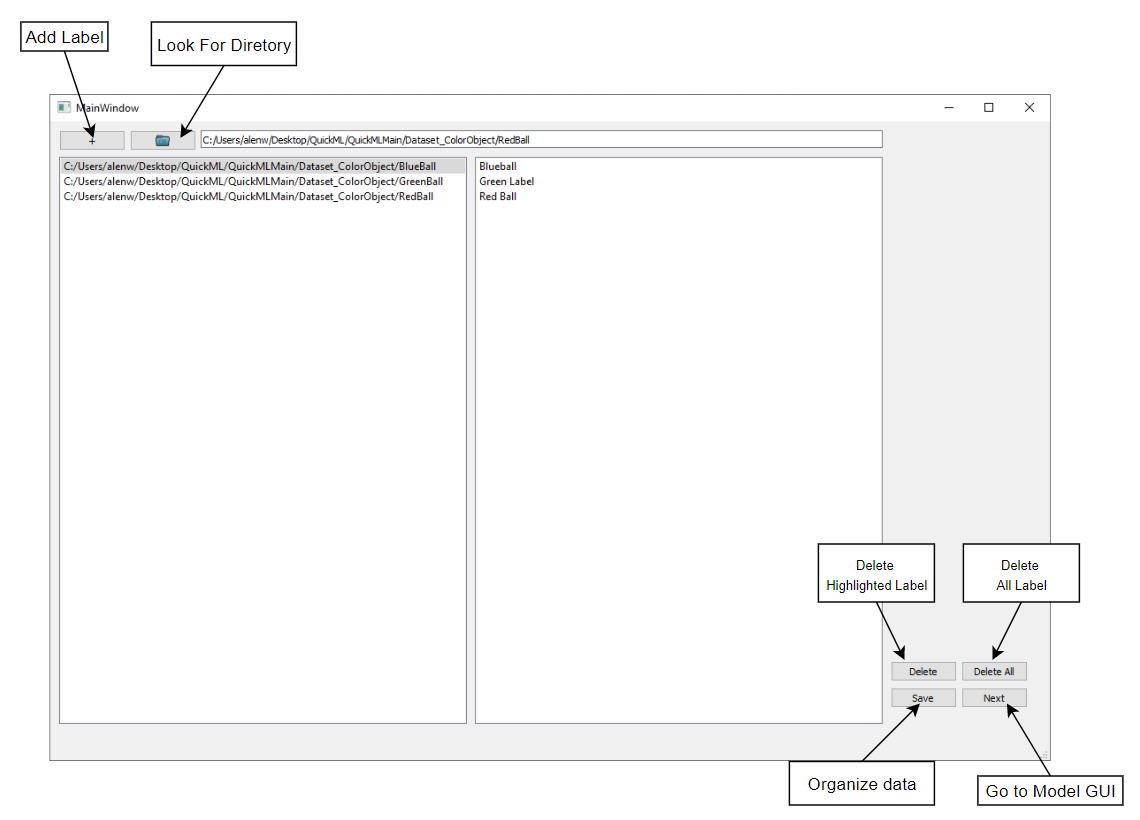
# Verification

1. Software verification and validation are the activities to check the software with its specification. All software projects produced must go through the process of verification and validation. (M. Jones, 1995) Developers typically do this by:
   1. Checking that the software meets their specified requirements.
   2. Checking each software item before they use it as an input to another software.
   3. Ensuring that the checks are done by other users besides the developer.
   4. Ensure that the amount of verification and validation effort suffices to show each software item in minimum operational use.

No matter the scale of a software project, software verification and validation have a large impact on a software quality. Software’s that do not go through this process are most likely to fail or not make it into production for its errors. According to the ESA Board, typically there are 20 to 50 errors per 1000 lines of code that occur during a software development cycle (M. Jones, 1995). All these errors that pile up can lead to operational failures and not meet specification requirement. Through Software verification and validation, we can reduce software errors to acceptable specifications.

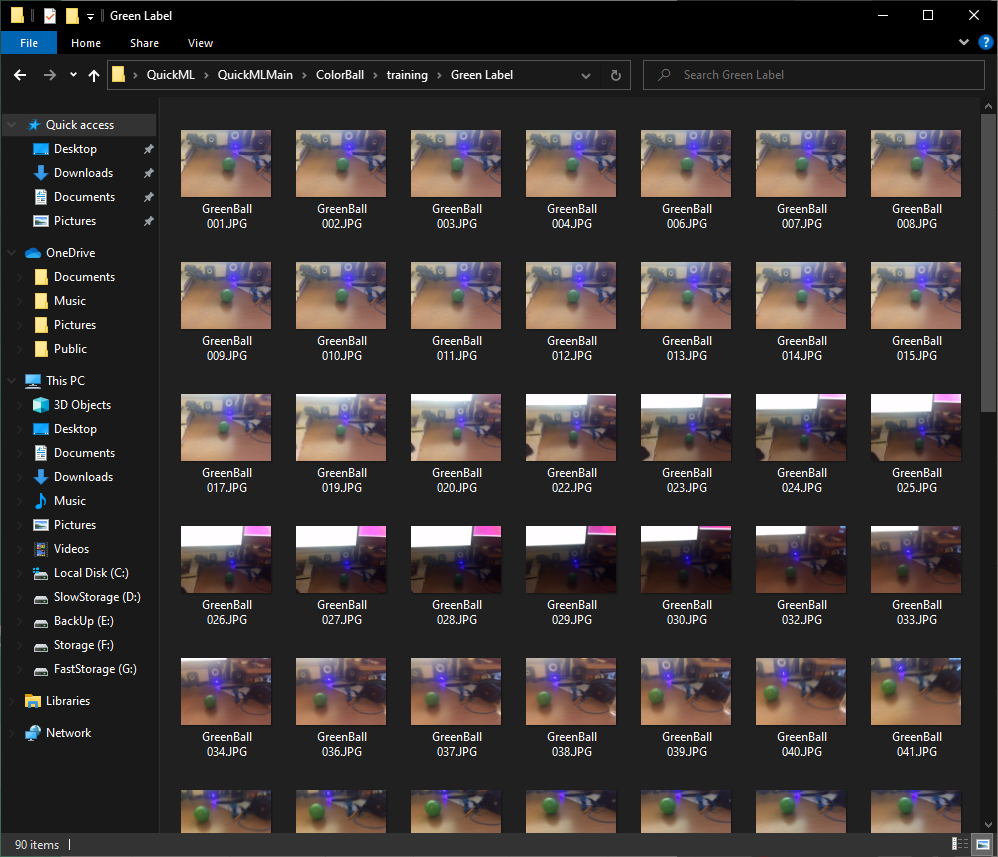
1. Our team verification and validation process are through testing each GUI individually, meaning we must test the Preprocessing GUI, Model GUI, and Testing GUI. Our team does this by using already made files and data to test each of the GUI individually. Once we have made sure that each GUI works as intended, in this way we can see how to structure data and files to pass it to the next GUI and run smoothly. Then we will test it at the system level, meaning we will test the entire software.
2. System Level test case

## Preprocessing GUI:

Here we are organizing some image datasets of colored objects and giving it arbitrary label names.

As you can see, the Preprocessing GUI shows the labels and are right next to their corresponding directory of where the images are located. When clicking the “save” button, an Input dialog will pop up and you will insert the project’s name. In our case, we just called it ColorBall.

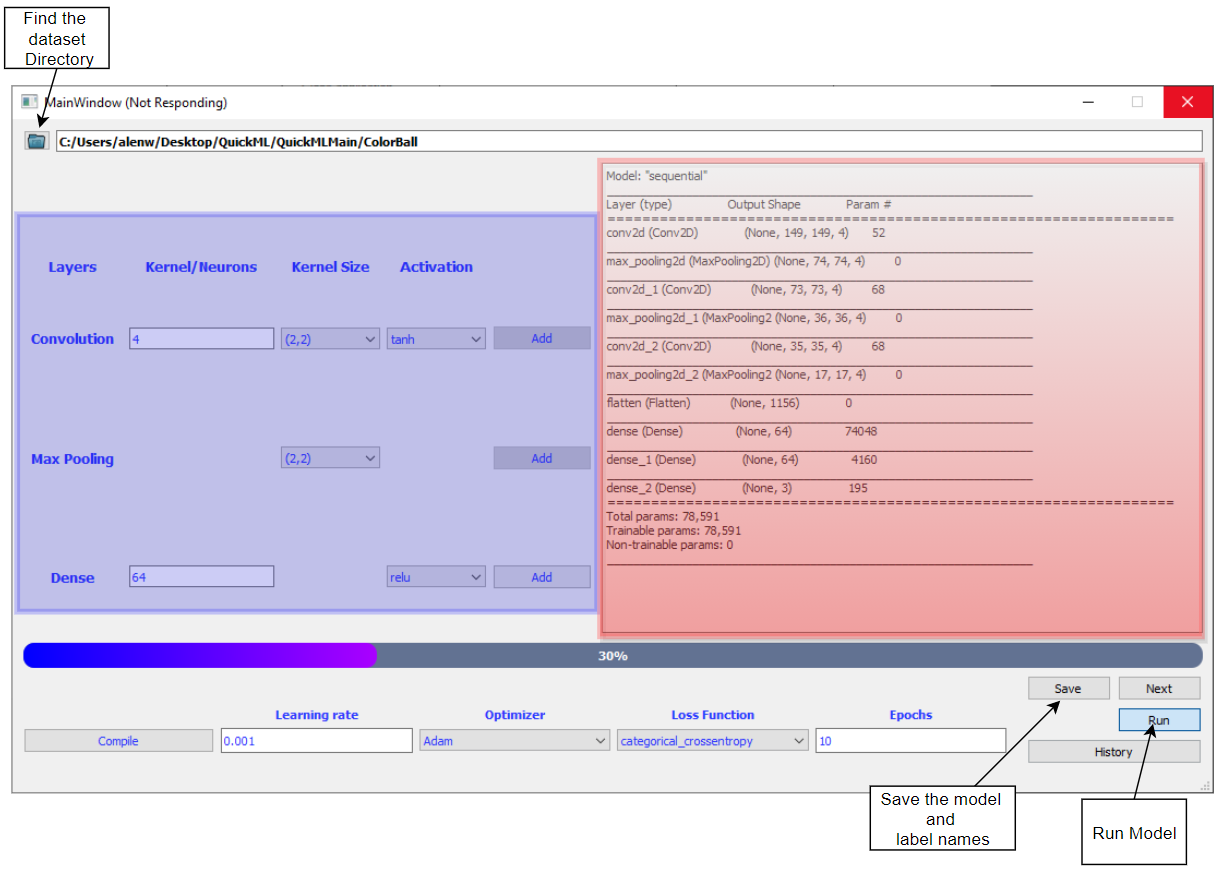
|  |  |  |  |
| --- | --- | --- | --- |
| Epic | User Story | Acceptance Criteria | Tested |
| 1. Preprocessing GUI | 1.1) As a user, I want to organize data to get it ready for the training step. | Verify that the user can just pick the folder with the images and give it a label name and is structured correctly for training step. | Checkmark |
| 1.2) As a user, I want to see the labels to know I put the right label name. | Verify that the user can see the labels on a text box and have the functionality to delete a label and its corresponding directory deleted. | Checkmark |

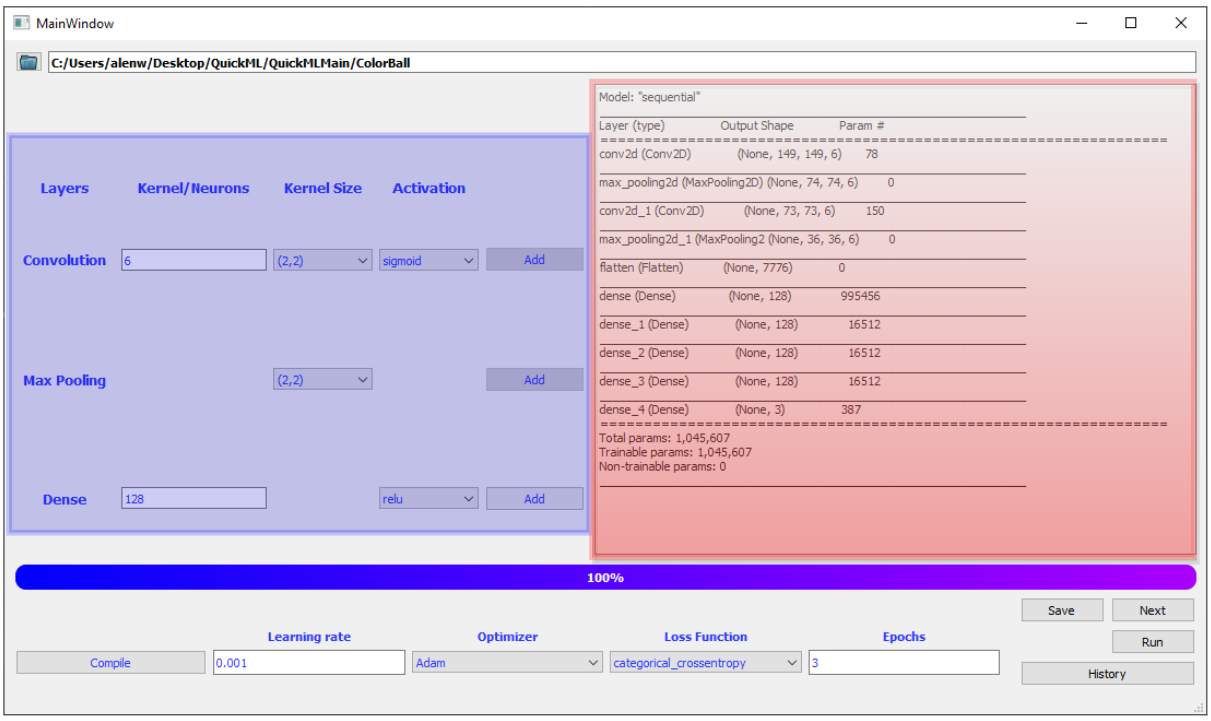
Now in the following image we can see that the GUI work as intended by creating an organized directory to pass it to the Model GUI to get it ready for training.

We used multiple types of image directories for this process. Our team used images of augmented images of hands that resembled the Rock, Paper, Scissors game. We also did it with images of cats and dogs and had to problem.

The current problem we are only experiencing in this GUI is that you can add Empty name labels or accidently add labels without a directory path. But this can easily be deleted by using the delete buttons. Technically, if the user were to proceed with the software like so it might cause software instability in the next GUIs, but ideally, we expect users to not cause such a noticeable error. This problem can be fixed though.

## Model GUI:

Here we have a snipper of the Model GUI running. As you can see through the progress bar, the model is training. The blue box represents the area where the user will be inputting parameters for the model. The red area are the results after the user has added a layer into the model.

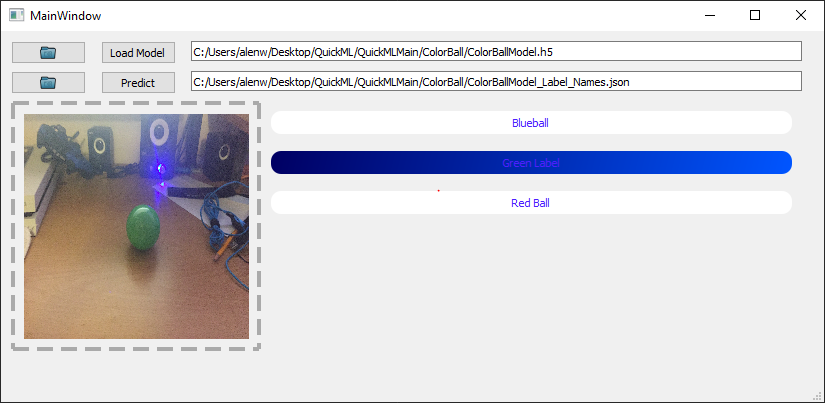
Here is one more test on the Model GUI with a different configuration.

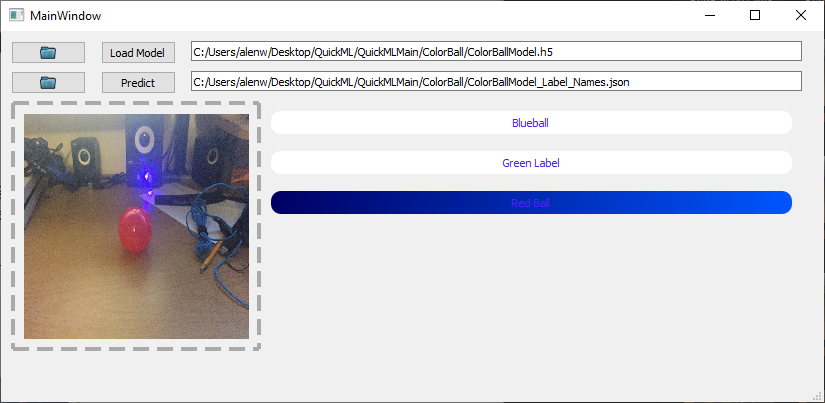
Initially, the Model GUI had many errors for inputs and other configuration that can cause problems and crashes when certain parameters were placed. Such as when there is no text in any of the text box in the blue box area, there was also the case of running or saving the model without an actual model. Most of this error causes the GUI to crash, but either eventually fixed it by disabling the buttons until a model is compiled or some background process were added to prevent the GUI from crashing.

|  |  |  |  |
| --- | --- | --- | --- |
| Epic | User Story | Acceptance Criteria | Tested |
| 2. Model GUI | 2.1) As a user, I want to make different configuration to see the effects of the model. | Verify that the user can make different configurations to make a Machine learning model | Checkmark |
| 2.2) As a user, I want to see the progress of the training so I can know how the process is going. | Verify that the progress bar in the GUI slowly increases as training is happening. | Checkmark |

## Testing GUI:

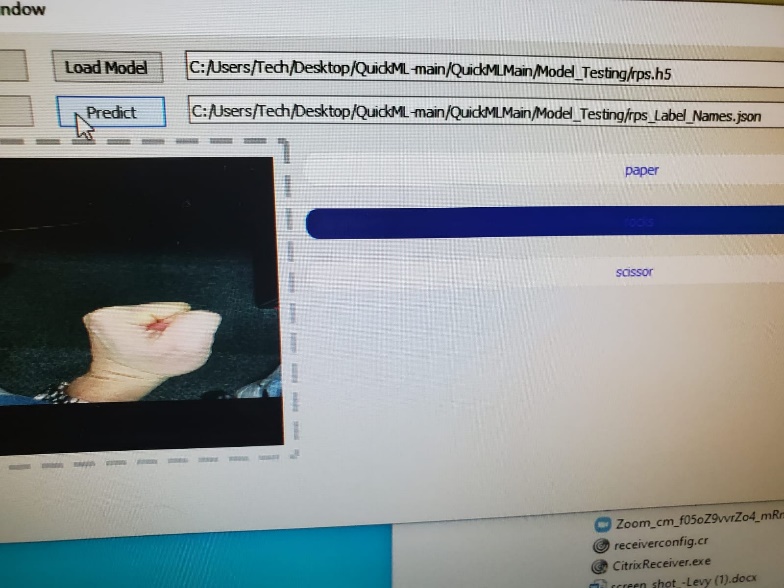
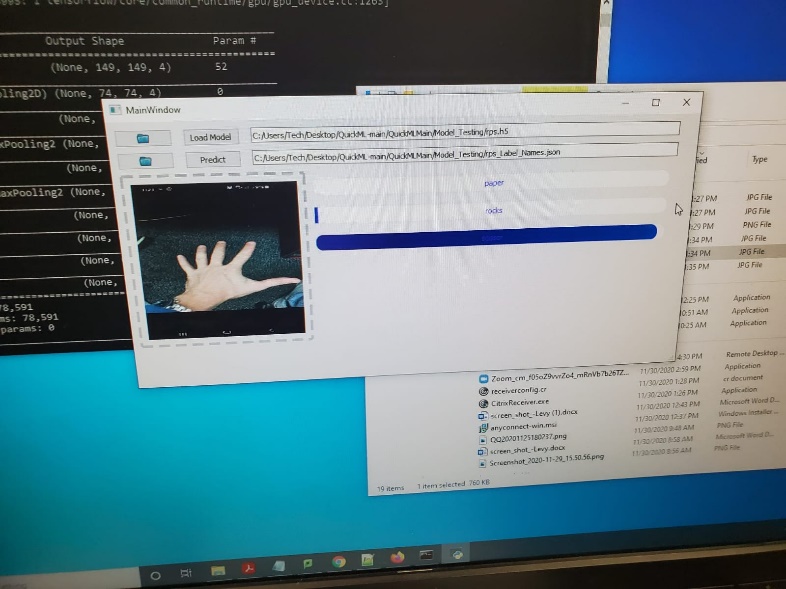
For the testing GUI we implemented a drag-and-drop box to input the image for ease of use and the blue bar tells us what type of image it is. The only problem we experience was predicting before loading the model, but that was easily fixed by disabling the predict button until the load model successfully run.





|  |  |  |  |
| --- | --- | --- | --- |
| Epic | User Story | Acceptance Criteria | Tested |
| 3. Model Testing GUI | 3.1) As a user, I want to load my model to test if the training worked. | Verify that the user can put a model file and be able to load the model to the GUI. | Checkmark |
| 3.2) As a user, I want to pass an image in the GUI and get a result. | Verify that the user has a way to input or drop an image to predict the image. | Checkmark |

* 1. Our team has also let other users that have experience with Machine Learning try out the program to see if they can figure out how to use it and these some results:



Some users told us some problems that were happening, so we had to guide them to have it run and that’s how we found some errors that cause the program to crash. But even with these errors they could just restart the GUI program, skip to the steps where they left off and could fix the mistakes to not cause the program to crash.

# Reflection

1. Thony Yan-Programmer / Machine Learning Engineer

In this project, I learn the software development cycle through the Agile method. Typically, in my past projects I would do the Waterfall style of software development since it was usually just doing the programming project individually. So, this was an exceptional learning experience on developing a software with Agile as a team, as I believe is more efficient and as a team, we can catch more errors during development and suggest fixes along the way.

This was also a great learning experience in creating GUI application. As we move forward in society, I am sure these skills will be valuable and knowing how to apply libraries and functions in programming to help others learn as critical to grow as a society. Specially with Machine Learning as it will be a prominent field in Computer Science and can solve things that typically human can’t.

1. Juan Lopez–Programmer / GUI Designer

During this project I’ve learned a lot about working in a two-person team with a deadline and GUI making for a neural network in python. Although we had started as a three-person team, soon it had become a two-person team. The project selection process was very swift. We had gone with a neural network in python, as neural networks are becoming more popular. The biggest challenge throughout the project was being able to fit in time for meeting together. As two full-time students and part-time workers, finding time to meet was very scarce. Yet we still pulled through and complete the project.

Throughout the entire project, my role was to create the GUI for the neural network. Anytime I had a question, I made sure to shoot it over to make sure I created the GUI the way we envisioned it. It also helped with my understanding of how this neural network works and how it was represented with the GUI. In total, there were 3 GUIs made. The most used tool for the actual look of the GUI was created with QTDesigner, which makes the form for the GUI. Every button, text field, image, and drop down was created first through QTDesigner to allow for the visual representation of the project. Fortunately, I already had some GUI making experience with NetBeans and their integrated GUI creator that had most of the same features. However, it was definitely a learning experience and the GUI itself is nothing compared to professionally made GUIs. Hopefully, in the future I can recreate this project once I have more experience with GUIs and neural networks.

Here is the timeline of our software development cycle. We had a rough start because of communication and a team member leaving, adding more pile of work for individuals. Although through the chart we see that we could not put as much time as planned, we could make a software that functions correctly. If we were to use the remaining work hours, we would of use that time to polish some GUIs and maybe give it a more appealing appearance.

# Bibliography

Daws, R. (2019, March 15). *Surprise! Machine learning jobs are high-paying and in-demand*. (AINEWS) Retrieved from https://artificialintelligence-news.com/2019/03/15/machine-learning-jobs-high-paying-demand/

M. Jones, U. M. (1995). *Guide To Software Verification and Validation.* Netherlands: ESA Publications Division.

Pierre Bourque, R. E. (2014). *Guide to the Software Engineering Body of Knowledge, Version 3.0.* IEEE Computer Society. Retrieved from www.swebok.org.

Terra, J. (2020, June 4). *The Rise of Artificial Intelligence and Machine Learning Job Trends*. (Simplilearn) Retrieved from https://www.simplilearn.com/rise-of-ai-and-machine-learning-job-trends-article