Microhistological Analysis of the Dietary Habits of Sheep

Arnav Gokhale, Godson Edewor,
Nate Scott, and Jack Nelson

CONCEPT OF OPERATIONS

REVISION – Final Draft

2 December 2024

CONCEPT OF OPERATIONS

FOR

Microhistological Analysis of the Dietary Habits of Sheep

TEAM 37

APPROVED BY:

_____

Project Leader Date

_____

Prof. Kalafatis Date

_____

T/A Date

Change Record

Rev.	Date	Originator	Approvals	Description
1.0	9/9/2024	GrazeView Application		Draft Release

Table of Contents

List of Tables

List of Figures

1. Execution and Validation Plan

Date Complete:	9/23/2024	9/30/2024	10/7/2024	10/14/2024	10/21/2024	10/28/2024	11/4/2024	11/11/2024	11/18/2024	11/25/2024	12/2/2024	12/5/2024	Due Date
Status Update 1													9/9/2024
Five Feature CNN Model													10/02/2024
Classify Data Sets (Grass & Image)													10/02/2024
U.I. Pathflow Complete									Complete				10/02/2024
User Manual Complete(Functionally)									In Progress				10/02/2024
Feature Testing									Not Started				10/16/2024
Image Splicer 50%													10/16/2024
Image Upload Complete													10/16/2024
Status Update 2													10/16/2024
50% Neural Net													10/30/2024
Automate New Data Entry to DB													10/30/2024
Loading Page Complete													10/30/2024
Five Feature Validation													11/13/2024
CSV (DB) Implementation complete													11/13/2024
Results Page Complete													11/13/2024
Background Design Complete													11/13/2024
Data Viewer Complete													11/20/2024
Expanded Data Viewer Complete													11/20/2024
User Manual Written													11/20/2024
Final Presentation													11/20/2024
Output Testing													11/26/2024
Design Resources Embedded													11/26/2024
Refine & Test for bugs (DB)													11/26/2024
Refine & Test for bugs (U.I.)													11/26/2024
Final Project Demo													11/26/2024
Final Report													12/05/2024

Table 1 : Execution Plan

Validation Plan

Paragraph #	Test Name	Success Criteria	Methodology	Status	Responsible Engineer(s)
9.2.1.1	Percentage Mapping	The FIve CNN can recognize, classify, and output percentages of the five unique grass features	Repeated validation and testing with different pictures of overlapping features	Completed	Godson Edewor
9.2.1.2	Fluid User Interface	All design components operate and are organized in a user-friendly fashion. Any and all resizing events are correctly handled, without layout errors.	Run the program and maneuver throughout the app using the design components, uploading image files, accessing the user manual, etc.	Completed	Jack Nelson
9.2.1.3	Imaging Database Pipeline	The DB can work with the model to input clean images, and store the output grass classifications	DB outputs normalized bounding box	Completed	Arnav Gokhale
9.2.1.4	Identification Time	The model operated through the user interface will process new images in less than 5 minutes	The runtime will be examined with a stop watch repeatedly	Completed	Godson Edewor
9.2.2.1	New Image Data Input(U.I)	New Data is added to DF	Create a test file and add it to the DF	Completed	Jack Nelson
9.2.2.2	Uploaded Data Storage	The application is successfully able to store data once the application is closed, and the application is successfully able to access this data once ran once more.	Create a folder within the project to store data. Using this folder, the data will be stored in a <List> format, in order to keep each upload's data separate from one another.	Completed	Jack Nelson
9.2.2.1.1	Five Feature Convolution Neural Network	The convolutional network will be built to distinguish between five grass features and hosted using Tensorflow's Keras	The Keras platform will return detailed observations on each run epoch of the five feature convolutional neural network	Completed	Godson Edewor
9.2.2.2.1	Interface Output Results	The Winforms Application will be able to take user input and generate sample data to be output, in a visually organized manner.	Create test methods that generate random ML data and save it with each sample upload	Completed	Jack Nelson
9.2.2.2.2	U.I. Help Guide	Each page will have access to a User Help Guide. Only one instance can be opened at one time.	Create a form that is only initialized/closed within other forms	Completed	Jack Nelson
9.2.2.2.3	Model Accuracy	The neural network will average 75% new image accuracy	Measure the success rate of vary network architectures and repeat until an optimal architecture is found then fit to new microscope images	Completed	Godson Edewor
9.2.4.1.1.1	CNN False Alarm	The Five CNN will not have a false alarm rate more than 10%	Use Keras to output detailed network statistics and repeatdly optimize it's error rate	Completed	Godson Edewor
9.2.1.2	Window Resizing Consistency	All design components are resized when the window is resized, background design included	Resize each window and confirm that each component stays at a reasonable size compared to the window	Completed	Nathan Scott
9.2.1.2	Component Testing	Each component (button, textbox, checkbox, etc.) works as needed	Each component has a function and purpose (no "useless" components)	Completed	Nathan Scott
9.2.1.3.4	Image Size Optimization	Model has both an efficient runtime and accuracy	Check between (50x50-500x500), no runtime or RAM errors	Completed	Arnav Gokhale
9.2.1.3.2	New Data Input Added to DB	All data is organized by grass type in CSV file	New files will be uploaded to DB, CSV file will be produced and verfied	Completed	Arnav Gokhale
9.2.1.2	Data Preview Windows	When uploading data, the window will include a preview window that shows the image before uploading	Upload several image files and see if the image will preview on the screen	Completed	Nathan Scott
9.2.1.3.3	Clean Image Verification	Model can run on and predict new images	Inspect images after "cleaning"	Completed	Arnav Gokhale
9.2.1.3.4	SQL Testing	DF stored in CSV and can be edited (new data)	Load the DataFrame into CSV, enabled for editing and adding new data.	Completed	Arnav Gokhale
9.2.1.2	All Design Resources Embedded	Can be run with local resources embedded within the project itself.	All design aspects (files, images, etc.) still appear when running the project executable file.	Completed	Nathan Scott

Table 2 : Validation Plan

2. Executive Summary

This project addresses the challenge of accurately analyzing the dietary habits of sheep through the development of a machine learning algorithm capable of processing microscopic images of sheep feces. Microhistological analysis, traditionally a labor-intensive task, involves the identification of plant fragments in stool samples to infer diet. By automating this process, the project aims to enhance both the efficiency and accuracy of dietary analysis, reducing the dependency on manual examination by experts.

The system being developed will use image processing and machine learning techniques to categorize different types of plant matter found in fecal samples. The machine learning algorithm will be trained on a diverse dataset of microscopic images to ensure robustness across different environmental and dietary variations, making it a versatile tool in agricultural fields.

The outcomes of this project will provide significant benefits to sheep farming and research communities. By automating the analysis of dietary intake, the tool will enable more timely and informed decisions regarding sheep nutrition, pasture management, and overall animal health. The reduction in manual labor will also lead to cost savings and increase the potential for wide-scale implementation across various settings.

This report outlines the key components of the project, including the technical operating concepts, system architecture, and the anticipated impacts. The system is expected to contribute to advancements in agricultural data analysis, offering a modern solution to an age-old challenge in livestock management.

3. Introduction

Microhistological analysis is a key method for studying the dietary habits of herbivores, including sheep. Traditional analysis requires expert intervention to identify plant fragments in fecal matter, which is time-consuming and often prone to errors. The development of machine learning algorithms provides an opportunity to automate and improve the accuracy of this process.

This project focuses on creating a machine learning system that can automatically analyze microscopic images of sheep stool samples to categorize plant matter. The system will integrate image processing techniques and advanced classification algorithms to provide a reliable and efficient tool for researchers and farmers.

3.1. Background

In sheep farming and research, understanding dietary habits is crucial for managing nutrition, health, and pasture resources. Traditionally, microhistological analysis has been the primary method for studying the diet of herbivores, including sheep. This process involves the manual identification of plant fragments in fecal samples using microscopic analysis. While effective, this method is highly labor-intensive, time-consuming, and requires trained personnel with expertise in plant morphology. Furthermore, the subjective nature of manual identification introduces the risk of human error, reducing the consistency and accuracy of the results.

The proposed system will enhance and potentially replace the traditional manual microhistological analysis by introducing a machine learning-driven solution that automates the identification and categorization of plant fragments in sheep stool samples. By utilizing image processing and classification algorithms, the system will process microscopic images and categorize plant fragments with higher speed and accuracy. This eliminates the bottleneck created by human labor and improves consistency across large-scale analyses.

Feature	**Enhancement**
Automation	The machine learning system significantly reduces the need for human intervention, allowing for faster processing of samples and enabling researchers and farmers to analyze dietary data in real-time
Consistency	While manual analysis is limited by time and expertise, the automated system can handle large datasets efficiently, making it suitable for use in both small-scale farms and research

	studies.
Cost Efficiency	By minimizing the reliance on trained personnel, the system reduces operational costs associated with dietary analysis while improving productivity.

Table 3: System Features

3.2. Overview

This project seeks to build a machine learning model capable of recognizing and classifying plant fragments in microscopic images of fecal matter. The system will be trained using a dataset of microscopic images and tested to ensure reliability. Once completed, this project will be able to have a user upload an image of sheep stool to the program, and the program will return a percentage breakdown of which/how much of a particular plant was in the stool sample.

3.3. Referenced Documents and Standards

The project adheres to recognized standards in machine learning and veterinary practices, ensuring ethical and technical guidelines are met. Key documents include industry standards on image classification.

4. Operating Concept

4.1. Scope

The GrazeView application proposed is designed for Texas A&M AgriLife in order to better identify and analyze stool samples collected from sheep in San Angelo, TX. This system is designed to increase the efficiency at which TAMU AgriLife can determine what diet is best for wool quality of said sheep.

4.2. Operational Description and Constraints

The user will first run the application to begin operation. From here, the user will choose to either upload new data(image only) or pre-analyzed data(image and text file). The image will need to be formatted as a .png file. Once uploaded, the data will be studied by the machine learning model, and then corresponding grass types and air bubbles will be identified.

Constraint	Reasoning
Data Storage	Due to this application being designed for any platform, data storage is limited to the capabilities of the users' laptop or PC. As well, each image file is fairly large, which limits which devices are able to operate the application effectively.
Machine Learning Accuracy	We are limited on the amount of time and training data we have to create a precise and accurate model. This will create limits on the systems' ability to identify both grass and air bubbles precisely.
Machine Learning Efficiency	Some image files that the machine learning model will have to analyze are 100mB and greater. Due to these images being this large, it may take the system more time to accurately study them.

Table 4: Constraints and Reasoning

4.3. System Description

The user-interface and machine learning model designs are shown below, in their respective subsystems.

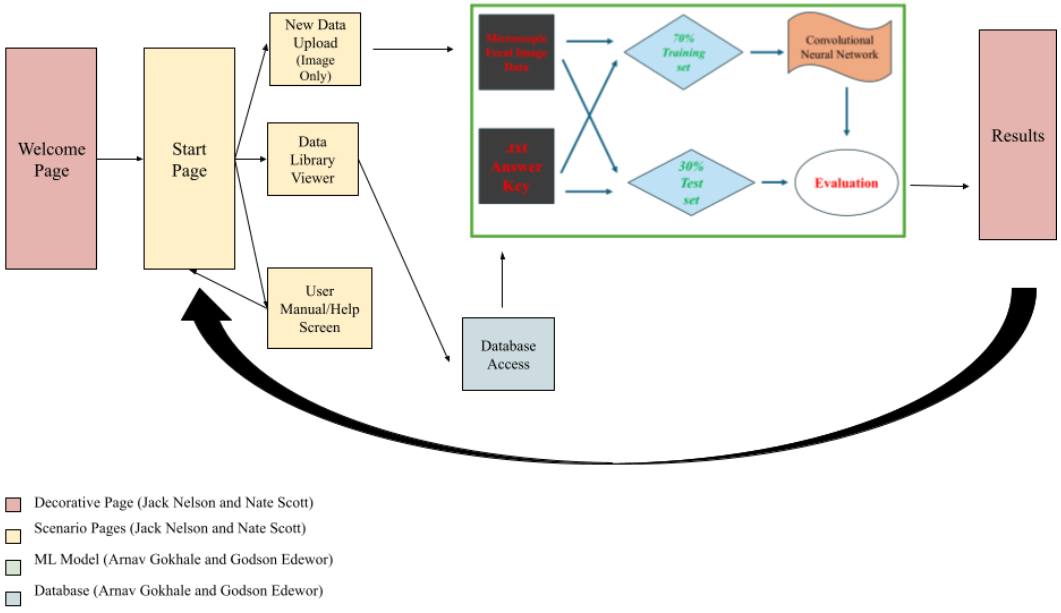

Figure 1 : Subsystem Flow Chart

4.4. Modes of Operations

With the requirements given by TAMU AgriLife, there will be two modes of operation, *New data Upload* and *Pre-Analyzed Data Upload*.

- *New Data Upload:* This mode of operation will allow the user to upload an image from the microscope, that neither the Machine Learning Model nor TAMU AgriLife have analyzed yet. The application will then analyze this new data and provide detailed results, based on the Model's current training.
- *View Data:* This mode of operation will allow the user to view all previous uploads within the application. This operation will provide each upload in a viewer-friendly fashion, with all data being passed correctly.

4.5. Users

This system is designed to be used by employees of TAMU AgriLife, specifically those currently researching the dietary habits of sheep. Employees should have a basic understanding of how each subsystem operates and how results are determined. The user interface will be designed in such a way that the user will not require any prior software development experience, only an understanding of web applications.

4.6. Support

User Help Guides will be provided in the following forms.

- *General User Guide:* A holistic user manual will be given within the application itself, and will provide step-by-step instructions on how to utilize each of the system's functions. Troubleshooting will also be given within this manual.
- *In-Source Documentation:* Within the source code itself, commentary will be provided in the case that the code itself needs to be modified. Commentary will be given in order to, at the very least, describe each function or method used.
- *References to Third-Party Documentation:* When necessary, the *General User Guide* will also provide references to external open software packages used.

5. Scenario(s)

5.1. Upload New Data

During this scenario, a new, unseen image (.png) file is uploaded to the system by the client. From there, the ML model will be able to analyze the new image and provide data on the image. The model will also be able to provide a result of the percentages of grass found in the image. For this scenario, the ML model will have needed to have been properly trained and tested to be able to analyze accurately and efficiently. Please refer to Figure 1 for a more detailed scenario process.

5.2. View Uploaded Data

During this scenario, the user will access the Data Library through the U.I., where all previous uploads will be stored and able to be viewed. This scenario will allow the user to sort data, preview the images from each upload, and export upload(s) into a printable format. As well, users will be able to clear data within this library, in order to help app performance, and as to not overload the application with data.

6. Analysis

6.1. Summary of Proposed Improvements

GrazeView will vastly reduce human error in sheep raising, save additional labor costs, and eliminate the need for expensive lab tests

- The model will be able to accommodate refitting with new data for higher precision, all while remaining non invasive on the sheep population
- Farmers will be empowered to create personalized feeding strategies to cater to each sheeps nutritional requirements
- The sheep model will be easily scalable to provide for the monitoring of both small and large flocks of sheep
- Herders will be able to easily predict whether a flock is experiencing health issues in advance by noticing abnormalities and/or irregularities in the model's output

6.2. Disadvantages and Limitations

Farmers in more rural areas may lack the necessary hardware, software, or internet connection required to run an advanced machine learning model such as GrazeView

- Re fitting GrazeView with new data is a long process that might not be cost effective for larger scale farms
- The model has a non zero chance of misinterpreting wrong grass types and can lack general understanding
- High initial implementation costs especially for farms with long standing practices or massive sheep flocks.

6.3. Alternatives

Substitutions for GrazeView do exist although they come at the expense of increased costs or slower implementation times

- Manually testing different grass diets on sheep through visual inspection, physical monitoring, record keeping, and data analysis
- Using crowdsourcing to leverage the efficiency of people to help classify different grass types for a specified cost
- Simple rule based/template matching to allow for simple classification of fecal grass types
- Consulting a veterinarian or agricultural professional to run high end laboratory/chemical analysis then installing personalized feeding systems to reduce the reliance on AI fecal analysis

6.4. Impact

Grazeview will help farmers process what pasture environments are best for their sheep as well as figure out ways to mitigate environmental impact from foreign grass types

- However when expanding to farmers with massive flocks or need for more fine tuned models, the energy consumption from training the model may negatively impact pro climate goals
- Development or disposal of onsite AI systems can bring ethical considerations especially for farmers who have a heavy reliance on an established ecosystem
- Rewiring a strong internet connection especially for more rural farmers or farms with on site servers will require expanding cellular infrastructure such as fiber optic cables, cell towers which may impact surrounding habitats or even the farm itself

Microhistological Analysis of the Dietary Habits of Sheep

Godson Edewor, Arnav Gokhale, Jack Nelson, and Nate Scott

FUNCTIONAL SYSTEM REQUIREMENTS

FUNCTIONAL SYSTEM REQUIREMENTS
FOR
Microhistological Analysis of the Dietary Habits of Sheep

PREPARED BY:

_____
Author Date

APPROVED BY:

_____
Project Leader Date

_____
John Lusher, P.E. Date

_____
T/A Date

Change Record

Rev.	Date	Originator	Approvals	Description
1.0	9-20-2024	GrazeView Application		Draft Release

7. Introduction

7.1. Purpose and Scope

This specification defines the technical requirements for the development items and support subsystems delivered to the client for the project focused on the automated analysis of sheep dietary habits through machine learning. The system aims to efficiently process microscopic images of sheep stool to categorize and analyze plant fragments, thus enhancing the understanding of sheep diets.

The scope of this document encompasses the necessary components and technical specifications required for successful implementation, including the integration of various subsystems such as Data Science, Machine Learning Model, User Interface Design, and User Interface Utility.

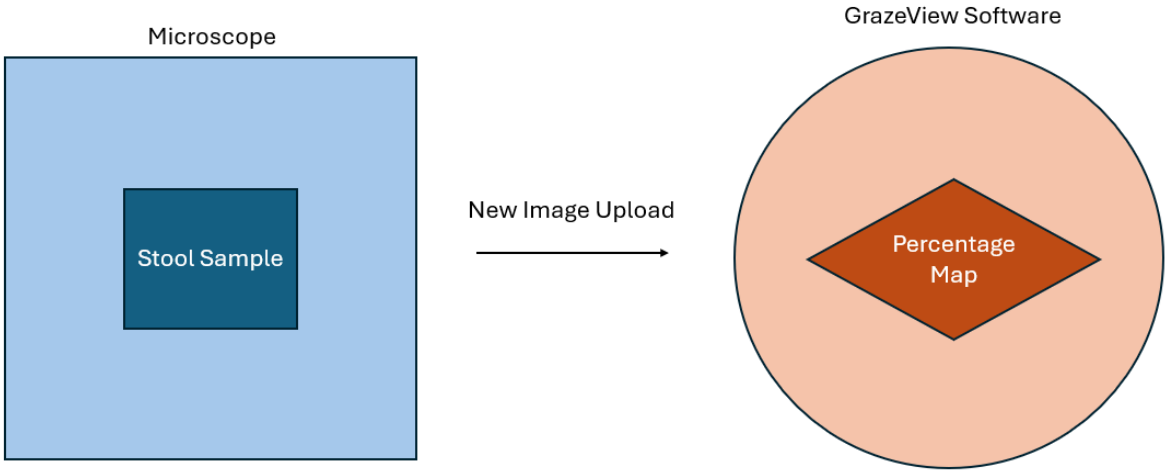

Figure 2. GrazeView Data Flow

7.2. Responsibility and Change Authority

Dr. Lusher is the sponsor of this project, and advises the team on how the project should be progressing. He also provided the team with the data, and which software to use in order to begin work on the project. Meetings occur between the team and either Dr. Lusher or a TA weekly, in order to ensure the team knows what they should be working on in order to deliver each subsystem by the end of the term.

8. Applicable and Reference Documents

8.1. Applicable Documents

The following documents, of the exact issue and revision shown, form a part of this specification to the extent specified herein:

Document Name	**Revision/Release Date**	**Publisher**
The Python Standard Library	3.12.6	Python Software Foundation

8.2. Reference Documents

The following documents are reference documents utilized in the development of this specification. These documents do not form a part of this specification and are not controlled by their reference herein.

Document Name	**Revision/Release Date**	**Publisher**
Desktop Guide (Windows Forms .NET)	.NET 8.0	Microsoft
TensorFlow Core	2.0	Google

8.3. Order of Precedence

In the event of a conflict between the text of this specification and an applicable document cited herein, the text of this specification takes precedence without any exceptions.

All specifications, standards, exhibits, drawings or other documents that are invoked as "applicable" in this specification are incorporated as cited. All documents that are referred to within an applicable report are considered to be for guidance and information only, except ICDs that have their relevant documents considered to be incorporated as cited.

9. Requirements

This section defines the minimum requirements that the GrazeView must meet. The requirements and constraints that apply to performance, design, interoperability, reliability, etc., of the system, are covered.

9.1. System Definition

GrazeView AI is a comprehensive analysis model trained to identify five major grass features that determine the dietary health of farm sheep. Using uploads of microscopic image data, GrazeView will be able to accurately classify the present grass types in any given slide and return relevant analysis for farmers, herders, and agricultural professionals to use in real world applications. The four main subsystems of the GrazeView AI are the Decorative Pages, Scenario Pages, Database Pipeline, and the Machine Learning Model

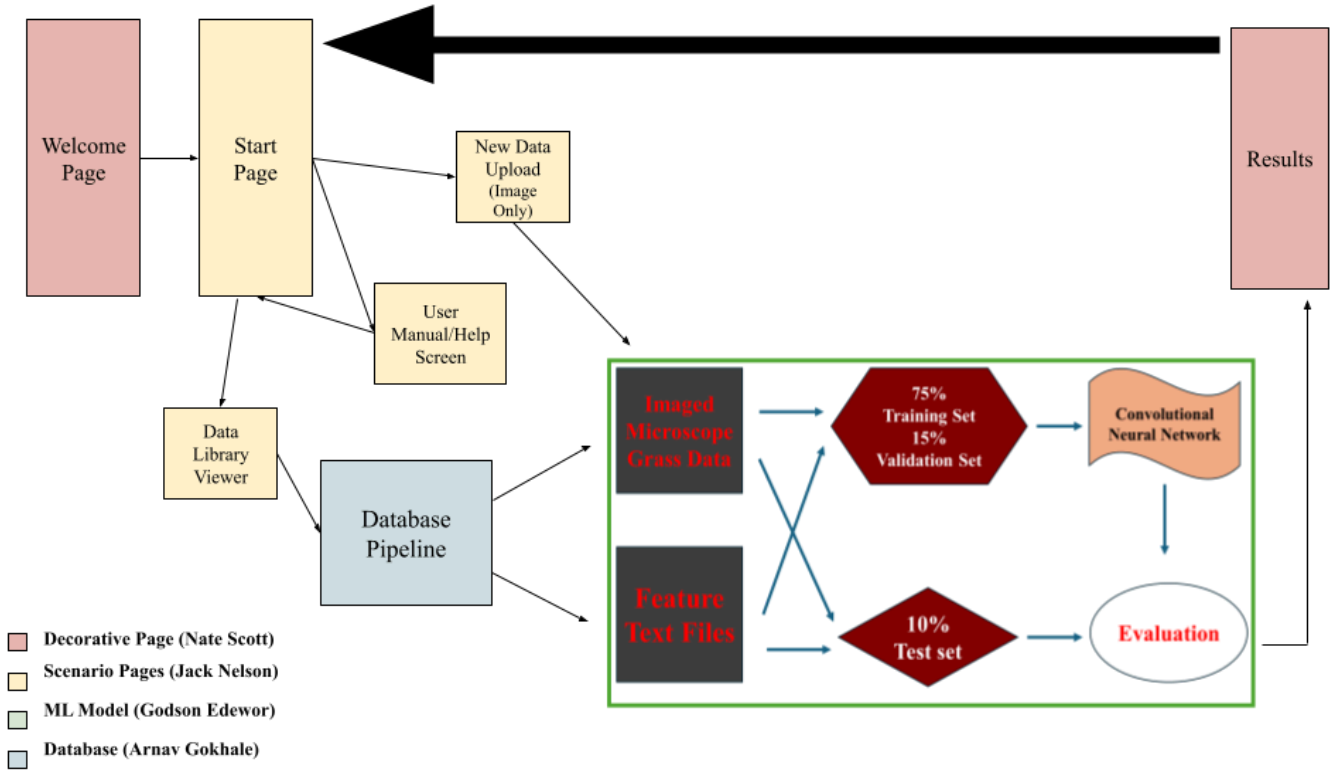

Figure 3. GrazeView System Diagram

Any user of the GrazeView application will first be directed to the welcome and start pages. From there they will be presented with a myriad of choices to either upload a new image, view the current data library, or access the preloaded user manual/help screen. Each of these will route the user to a

new page with different functionality. When the user chooses to upload a new image, each image will be first stored into the general data library then the python pipeline script will splice the images into even slides to be analyzed by the trained GrazeView convolutional neural network. The GrazeView ML model will then display an complete array of grass feature percentages for each available type of grass on the given slide, which will allow users to make detailed conclusions using GrazeView's results instead of having to manually identify each microscopic grass type on the thousands of possible slide images.

9.2. Characteristics

9.2.1. Functional / Performance Requirements

9.2.1.1. Percentage Mapping

GrazeView AI will create a map of percentages highlighting what grass types make up a percent of any given spliced slide. Each spliced slide percentage will then be tallied up to give a full picture of grass analysis for the user.

> *Rationale: Percentage cross-mapping is the quickest way of converting the AI's analysis results into valuable results given the size of the image database*

9.2.1.2. Fluid User Interface

User interactions with the GrazeView model will be through a fluid interface that is intuitively understood, interconnected with the backend neural network, and simple enough to be mass utilized by agricultural professionals nationwide

> *Rationale: The complexity of the model won't be exposed to the users as the UI will handle all user requests including edge cases that causes errors*

9.2.1.3. Imaging Database Pipeline

To both analyze new image data and create effective training, validation, and testing sets for the backend neural network, a database pipeline will be set up using a python script to scrape the text feature files and the images.

> *Rationale: The convolutional neural network will require large amounts of data that is unfeasible to manually input therefore an automatic pipeline will be implemented to supplement this*

9.2.1.4. Identification Time

The comprehensive time necessary to classify any given image will not exceed 40 minutes

> *Rationale: A constraint of GrazeView AI is the large sized images which easily exceed tens of thousands of pixels, therefore reducing the time required to analyze each image will be necessary for full user functionality*

9.2.2. Software Characteristics

9.2.2.1. New Image Data Input

a. Images uploaded through the user interface will be spliced into subsections before being scanned and analyzed by the neural network. A list of found feature percentages will be generated using the trained final model

b. No sequence of inputted user commands shall damage GrazeView or cause any malfunction.

Rationale: By design, the user interface/help guide should limit the chance of damage or malfunction by user/technician error

9.2.2.2. Uploaded Data Storage

a. Image files, user input, and ML data will be stored in a database within the application, that allows access to past data when the application is closed and reopened.
b. There will be no corruption or invalid access to this stored data.

9.2.2.2.1 Five Feature Convolutional Neural Network

Using the microhistological dataset and Tensorflow custom pipeline to establish training, validation, and testing sets, the user facing GrazeView AI model will comprise of a trained CNN that identifies the five separate grass types

Rationale: Tensorflow is the most developed and complete framework for making large scale generalizable neural networks

9.2.2.2.2 Operating System

The application shall run on Windows OS and will require at least 10 GB of RAM

Rationale: This is the optimal set up for analyzing the large image files using a reliable machine framework

9.2.2.3. Outputs

9.2.2.3.1 Interface Results

The feature percentage results of the trained model's analysis shall be displayed using the graphical framework Winforms in Visual studio

Rationale: Winforms offers GrazeView the widest applicability to be ran on Windows based systems

9.2.2.3.2 UI Help Guide

A user manual/help guide will be available at all times on the main page of GrazeView. It will contain how to use GrazeView, tips for troubleshooting the software, and to ensure effective analysis by the model.

Rationale: Provides the ability to control things for debugging manually and a way to view/download the node map with associated potential targets.

9.2.2.3.3 Model Accuracy

The use of Tensorflow to build the model will minimize both the risks of overfitting and the model algorithms will characterize the multiple features with less than 20% error

Rationale: Tensorflow is a stable and highly reliable framework capable of supporting the python packages to build Grazeview

9.2.3. Environmental Concerns

A trained GrazeView model has high energy demands which increases proportionally to the complexity of the model and how much data has been parsed. Tensorflow's sequential analysis algorithms will be used to mitigate this effect and improve accuracy.

Rationale: Decreasing the networks energy consumption will help GrazeView be more applicable for our users

9.2.4. Infrastructure Requirements

The application of GrazeView relies on our users having reliable cellular infrastructure to allow the model to be more regionally located to the local pastures which is especially important for users farther out in lesser developed rural areas.

Rationale: Grazeview shall be applicable enough to be operated by both professionals and standalone users

9.2.4.1. Failure Detection

9.2.4.1.1 Testing Training Sets

Each training, validation, and testing set will be evaluated before being processed by the database python pipeline script into custom datasets for the GrazeView convolutional model

Rationale: This is necessary as the model's accuracy correlates with the quality of the training data

9.2.4.1.1.1 CNN False Alarms

The CNN shall have a false alarm rate of less than 10 percent.

Rationale: This is a requirement necessary for the users due to constraints of the microscope in which the GrazeView AI is integrating.

9.2.4.1.1.2 Epochs Error Log

Error and network updates statistics will be stored to allow users to further analyze the model's percentage outputs epoch by epoch

Rationale: GrazeView's comprehension statistics will be openly available to our users

10. Support Requirements

The GrazeView application will be solely software based, and as such, will not have any physical support once implemented. Troubleshooting techniques will be given within the application, but will be left to the user to resolve.

10.1. User Requirements

The GrazeView application requires a Windows OS machine with a minimum of 10 GB of RAM, 10 GB of free disk space, and a modern multi-core processor. Users must provide this device, where they will then be able to download and install the application. As well, users must have a basic understanding of the operation of their Windows device.

10.2. Application Requirements

Within the application itself, a *User Guide* will be included, which will include operational instruction of the application, as well as troubleshooting support. As well, this *User Guide* will reference third-party external packages used within the program, should any issues arise from the program itself. The program will include documentation of methods and techniques used, in order to allow the user to easily adjust code if necessary.

1. Appendix A: Acronyms and Abbreviations

API	Application Program Interface
BIT	Built-In Test
GUI	Graphical User Interface
ICD	Interface Control Document
ML	Machine Learning
CNN	Convolutional Neural Network

2. Appendix B: Definition of Terms

1. *Application Program Interface(API)* : A set of protocols and tools for building and integrating software applications.
2. *Cross-Validation* : A technique for assessing how well an ML Model generalizes new data by splitting the data into multiple subsets.
3. *Graphical User Interface (GUI)* : A visual interface through which users interact with the WinForms Application.
4. *Machine Learning (ML)* : A subset of artificial intelligence(AI) that focuses on building systems that can learn from and make decisions based on data, without explicit programming for each task.
5. *Model* : An abstraction of the machine learning algorithm that makes predictions or decisions based on data.
6. *Neural Networks :* A series of algorithms in machine learning modeled after the human brain.
7. *Supervised Model :* A ML Model that is trained on labeled datasets. The Model is given the problem and solution, and must learn to map them correctly.
8. *Test Data :* A dataset unseen by the model during training.
9. *Training Data :* A dataset used to train the machine learning model.
10. *Unsupervised Model :* A ML Model that is trained on unlabeled datasets.
11. *Windows Application/Winforms :* A software program that runs on Microsoft Windows operating system, designed with a graphical user interface(GUI) for user interaction.

Microhistological Analysis of the Dietary Habits of Sheep

Godson Edewor, Arnav Gokhale, Jack Nelson, and Nate Scott

Interface Control Document

INTERFACE CONTROL DOCUMENT
FOR
Microhistological Analysis of the Dietary Habits of Sheep

PREPARED BY:

_____
Author Date

APPROVED BY:

_____
Project Leader Date

_____
John Lusher II, P.E. Date

_____
T/A Date

Change Record

Rev.	Date	Originator	Approvals	Description
1.0	9-20-2024	GrazeView Application		Draft Release

11. Overview

This Interface Control Document (ICD) provides a comprehensive overview of the interfaces and interactions between the various subsystems involved in the project. It defines the essential protocols, data structures, and design principles that will ensure effective communication and integration among the software components.

3. References and Definitions

This section includes critical reference documents and definitions relevant to the project, such as the Desktop Guide and TensorFlow Core specifications, as well as key terms like Application Program Interface (API), Machine Learning (ML), and User Interface (UI) concepts.

4. Overall Interface Principles and Considerations

The project interfaces are entirely software-based, focusing on principles that promote consistency and simplicity in user interactions. User experience design principles emphasize professional presentation and clear feedback to users.
The section covers error handling and validation strategies to ensure data integrity, providing users with meaningful error messages and allowing continued interaction with the application despite errors.

5. User Interface Operations

The GrazeView application will facilitate three primary operations for users:
View User Guide: Accessing the user guide for assistance.
Upload New Data: Users can upload image files for analysis by the ML model, with real-time loading indicators to track progress.
Upload Training Data: Users can upload image and text files to enhance the ML model's training, also providing feedback during the process.

6. Application to Machine Learning Model Communication

The WinForms application will interface with the ML model through backend C# code that manages the transfer of data. This section details how the application handles user inputs, communicates with TensorFlow for analysis, and presents results back to the user interface.

7. Analysis API Interface

This section includes examples of API endpoints for uploading new and pre-analyzed training data, facilitating communication between the user interface and machine learning backend.

8. Communications / Device Interface Protocols

The ICD will also outline the necessary protocols for communication, such as wireless communication standards (e.g., IEEE 802.11ac), host device interfaces (e.g., USB 3.0), and any custom protocols for device peripherals.

12. References and Definitions

12.1. References

Document	Revision/Release Date	Publisher
Desktop Guide(Windows Forms .NET)	.NET 8.0	Microsoft
TensorFlow Core	2.0	Google

Table 5 : Reference Documents

12.2. Definitions

Application Program Interface(API) : A set of protocols and tools for building and integrating software applications.

Cross-Validation : A technique for assessing how well an ML Model generalizes new data by splitting the data into multiple subsets.

Graphical User Interface (GUI) : A visual interface through which users interact with the WinForms Application.

Machine Learning (ML) : A subset of artificial intelligence(AI) that focuses on building systems that can learn from and make decisions based on data, without explicit programming for each task.

Model : An abstraction of the machine learning algorithm that makes predictions or decisions based on data.

Neural Networks : A series of algorithms in machine learning modeled after the human brain.

Supervised Model : A ML Model that is trained on labeled datasets. The Model is given the problem and solution, and must learn to map them correctly.

Test Data : A dataset unseen by the model during training.

Training Data : A dataset used to train the machine learning model.

Unsupervised Model : A ML Model that is trained on unlabeled datasets.

Windows Application/Winforms : A software program that runs on Microsoft Windows operating system, designed with a graphical user interface(GUI) for user interaction.

13. Overall Interface Principles and Considerations

The interfaces for this project are solely software based, and therefore no physical interface applies. We have defined several principles and considerations to be applied in order to keep subsystems consistent.

13.1. User Experience Design Principles

The interface design will be created to ensure both consistency and simplicity, in order to avoid user confusion. In using WinForms to design the U.I., clear feedback will also be created in order to keep the user updated on what is happening. The page design itself will reflect professionalism, and ease of use.

13.2. Error Handling and Validation

The user interface will enforce input validation, ensuring that users are correctly uploading data, prior to use of the TensorFlow model. Errors within the model will also be returned to the user interface, using messages or prompts that explain the problem and suggest possible solutions. As well, the application will be designed as such to allow users to continue with other features, should the model encounter errors.

13.3. Integration with TensorFlow

The data input/output interface will be designed to allow users to input data as images or text files, with the option to use as training data or as new data. The model feedback will be designed in a user-friendly way, showing clear predictions and progress updates as the data is analyzed. For real-time processing, the application will also provide loading indicators to keep users informed.

14. User Interface

14.1. User ↔ Application

The GrazeView Application will provide the user with three different operations:

- *View User Guide:* This operation will exit the main page and bring the user to the *User Guide*, outlined in ***4.6 Support***.
- *Upload New Data:* This operation will allow the user to upload an image file(.png, .jpg, .jpeg, etc.) that will be analyzed by the ML model as Test Data. The user will be able to watch the progress of the model via loading indicators, and once complete, the application will output the analyzed data to be viewed by the user.
- *View Data:* This operation will allow the user to view all saved uploads, where they can sort, view, print, or clear data within the library.

14.2. Application ↔ Machine Learning Model

The WinForms application will communicate with the Machine Learning(ML) model through backend C# code that connects the user interface(UI) with TensorFlow. Once prompted, the user will upload either an image file or image and text files, which will then be sent to the ML model to be analyzed. This backend C# programming will allow the data to be then returned to the UI, once analyzed. From there, results will be shown visually and graphically within the application's UI.

15. Analysis API Interface

See example:

15.1.A (Upload New Data)

Endpoint: /api/v1/upload_new_data
Method: POST
Description: This API endpoint allows users to upload image files (.png) that will be analyzed by the machine learning model.

Input:
Parameters:
image_file: The raw image file to be analyzed.
image_id: A unique identifier for the uploaded image.

Output:
Response:
status: Confirmation of the upload and processing status (e.g., success/failure).
analysis_result: The results of the analysis, including categorized plant fragments and associated probabilities.

15.2.B (Upload Pre-Analyzed Training Data)

Endpoint: /api/v1/upload_training_data
Method: POST
Description: This API endpoint allows users to upload both image files and corresponding text files (e.g., annotations) to provide additional training data for the machine learning model.

Input
Parameters:
image_file: The image file to be used as training data.
text_file: The associated text file containing labels or annotations for the training data.
training_id: A unique identifier for the training dataset.

Output

Response:

status: Confirmation of the upload and processing status (e.g., success/failure).

training_feedback: Feedback on the training process, including any improvements in model accuracy.

Microhistological Analysis of the Dietary Habits of Sheep

Godson Edewor, Arnav Gokhale, Jack Nelson, and Nate Scott

SCHEDULE

Execution Plan for Microhistological Analysis of the Dietary Habits of Sheep

Date Complete:	9/23/2024	9/30/2024	10/7/2024	10/14/2024	10/21/2024	10/28/2024	11/4/2024	11/11/2024	11/18/2024	11/25/2024	12/2/2024	12/5/2024	Due Date
Status Update 1													9/9/2024
Five Feature CNN Model													10/02/2024
Classify Data Sets (Grass & Image)													10/02/2024
U.I. Pathflow Complete									Complete				10/02/2024
User Manual Complete(Functionally)									In Progress				10/02/2024
Feature Testing									Not Started				10/16/2024
Image Splicer 50%													10/16/2024
Image Upload Complete													10/16/2024
Status Update 2													10/16/2024
50% Neural Net													10/30/2024
Automate New Data Entry to DB													10/30/2024
Loading Page Complete													10/30/2024
Five Feature Validation													11/13/2024
CSV (DB) Implementation complete													11/13/2024
Results Page Complete													11/13/2024
Background Design Complete													11/13/2024
Data Viewer Complete													11/20/2024
Expanded Data Viewer Complete													11/20/2024
User Manual Written													11/20/2024
Final Presentation													11/20/2024
Output Testing													11/26/2024
Design Resources Embedded													11/26/2024
Refine & Test for bugs (DB)													11/26/2024
Refine & Test for bugs (U.I.)													11/26/2024
Final Project Demo													11/26/2024
Final Report													12/05/2024

Performance on Execution Plan

The individual subsystems all completed their execution planning that was outlined for the proposed problem. Progress was kept up and updated on a week by week basis which allowed our objectives to be held accountable. Overall, the project is both in full completion, ready for the upcoming integration, as well as the app deployment.

Microhistological Analysis of the Dietary Habits of Sheep

Godson Edewor, Arnav Gokhale, Jack Nelson, and Nate Scott

VALIDATION

Validation Plan for Microhistological Analysis of the Dietary Habits of Sheep

Validation Plan

Paragraph #	Test Name	Success Criteria	Methodology	Status	Responsible Engineer(s)
9.2.1.1	Percentage Mapping	The FIve CNN can recognize, classify, and output percentages of the five unique grass features	Repeated validation and testing with different pictures of overlapping features	Completed	Godson Edewor
9.2.1.2	Fluid User Interface	All design components operate and are organized in a user-friendly fashion. Any and all resizing events are correctly handled, without layout errors.	Run the program and maneuver throughout the app using the design components, uploading image files, accessing the user manual, etc.	Completed	Jack Nelson
9.2.1.3	Imaging Database Pipeline	The DB can work with the model to input clean images, and store the output grass classifications	DB outputs normalized bounding box	Completed	Arnav Gokhale
9.2.1.4	Identification Time	The model operated through the user interface will process new images in less than 5 minutes	The runtime will be examined with a stop watch repeatedly	Completed	Godson Edewor
9.2.2.1	New Image Data Input(U.I)	New Data is added to DF	Create a test file and add it to the DF	Completed	Jack Nelson
9.2.2.2	Uploaded Data Storage	The application is successfully able to store data once the application is closed, and the application is successfully able to access this data once ran once more.	Create a folder within the project to store data. Using this folder, the data will be stored in a <List> format, in order to keep each upload's data separate from one another.	Completed	Jack Nelson
9.2.2.1.1	Five Feature Convolution Neural Network	The convolutional network will be built to distinguish between five grass features and hosted using Tensorflow's Keras	The Keras platform will return detailed observations on each run epoch of the five feature convolutional neural network	Completed	Godson Edewor
9.2.2.2.1	Interface Output Results	The Winforms Application will be able to take user input and generate sample data to be output, in a visually organized manner.	Create test methods that generate random ML data and save it with each sample upload	Completed	Jack Nelson
9.2.2.2.2	U.I. Help Guide	Each page will have access to a User Help Guide. Only one instance can be opened at one time.	Create a form that is only initialized/closed within other forms	Completed	Jack Nelson
9.2.2.2.3	Model Accuracy	The neural network will average 75% new image accuracy	Measure the success rate of vary network architectures and repeat until an optimal architecture is found then fit to new microscope images	Completed	Godson Edewor
9.2.4.1.1.1	CNN False Alarm	The Five CNN will not have a false alarm rate more than 10%	Use Keras to output detailed network statistics and repeatdly optimize it's error rate	Completed	Godson Edewor
9.2.1.2	Window Resizing Consistency	All design components are resized when the window is resized, background design included	Resize each window and confirm that each component stays at a reasonable size compared to the window	Completed	Nathan Scott
9.2.1.2	Component Testing	Each component (button, textbox, checkbox, etc.) works as needed	Each component has a function and purpose (no "useless" components)	Completed	Nathan Scott
9.2.1.3.4	Image Size Optimization	Model has both an efficient runtime and accuracy	Check between (50x50-500x500), no runtime or RAM errors	Completed	Arnav Gokhale
9.2.1.3.2	New Data Input Added to DB	All data is organized by grass type in CSV file	New files will be uploaded to DB, CSV file will be produced and verfied	Completed	Arnav Gokhale
9.2.1.2	Data Preview Windows	When uploading data, the window will include a preview window that shows the image before uploading	Upload several image files and see if the image will preview on the screen	Completed	Nathan Scott
9.2.1.3.3	Clean Image Verification	Model can run on and predict new images	Inspect images after "cleaning"	Completed	Arnav Gokhale
9.2.1.3.4	SQL Testing	DF stored in CSV and can be edited (new data)	Load the DataFrame into CSV, enabled for editing and adding new data.	Completed	Arnav Gokhale
9.2.1.2	All Design Resources Embedded	Can be run with local resources embedded within the project itself.	All design aspects (files, images, etc.) still appear when running the project executable file.	Completed	Nathan Scott

Performance on Validation Plan

The validation plan was created primarily by taking into consideration the upcoming subsystem integration of each individual subsystem. So prior to integration, each subsystem was tested for errors and each individual step of assembly was validated to ensure a seamless dataflow from Database to Machine Learning to U.I. Web Application. The validation methodologies also helped to ensure that the GrazeView App would be more reliable for our users upon deployment.

Microhistological Analysis of the Dietary Habits of Sheep

Godson Edewor, Arnav Gokhale, Jack Nelson, and Nate Scott

SUBSYSTEM REPORTS

SUBSYSTEM REPORTS
FOR
Microhistological Analysis of the Dietary Habits of Sheep

PREPARED BY:

_____
Author Date

APPROVED BY:

_____
Project Leader Date

_____
John Lusher, P.E. Date

_____
T/A Date

Change Record

Rev.	Date	Originator	Approvals	Description
1.0	12-02-2024	GrazeView Application		Draft Release

16. Machine Learning Subsystem - Godson Edewor

16.1. Subsystem Introduction

For this capstone semester project, the GrazeView Model was developed specifically to process and analyze the inputted microscopic grass feature images for use in real world rural farm applications. With the provided TAMU AgriLife Dataset, each feature image is spliced into its corresponding sub feature image then transformed into a series of RGB Alpha array vectors. The multilayered convolutional neural network is then fitted to first predict the four possible unique features, nale grass, qufu grass, erci grass, and air bubbles from the given subimages and second output a list of percentages classifying the present features in any given microscope slide. Finally, a multitude of machine learning improvement techniques are applied to ultimately ensure the model's practicality for both academic research purposes and client facing industry implementations.

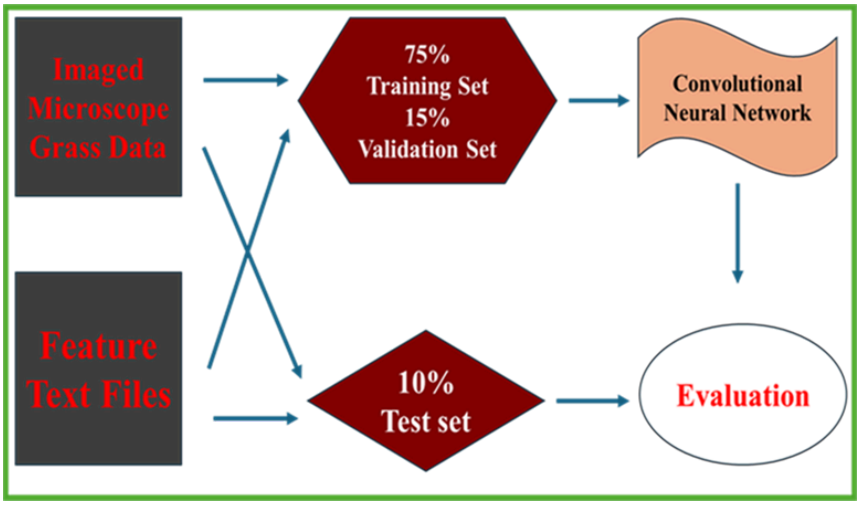

Figure 4: Four Feature CNN Flowchart

16.2. Sub-Image Splicing

16.2.1. Feature Splicing

Firstly to give the model a more comprehensive and detailed initial inspection of the provided grass feature data from TAMU AgriLife, each feature image of the four respective classes were spliced into 30 - 40 subimages depending on the size of the specific image. This allowed the previously limited data to be expanded by over ten times the original size of the feature dataset and gave the model more leeway to make complex decisions. To filter out unnecessary noise, a whitespace filter was embedded in the splicing python script that analyzed the RGB and alpha values of each sub image to remove images that contained over 95% whitespace. The python script itself was created using the MatPlotLib libraries to process each text file, snip the feature images with the provided dimensions, then further refine the feature images into detailed spliced subimages. A new sub image feature dataset was created and primarily used to train the convolutional neural network.

16.2.2. Validation

Prior to implementing the spliced sub image technique, a rudimentary model was built and tested

on the provided original unspliced feature image dataset. As the images were too few and too similar for the network to extrapolate constructive conclusions, the overall validation accuracy plateaued to 60%. This created the necessity for the dataset to be expanded and the model to focus on more cellular features rather than full length grass particles. Thus, implementing the feature splicing massively improved the accuracy in the long run and constructed a more diversified model that had room to be applied to even more additional features.

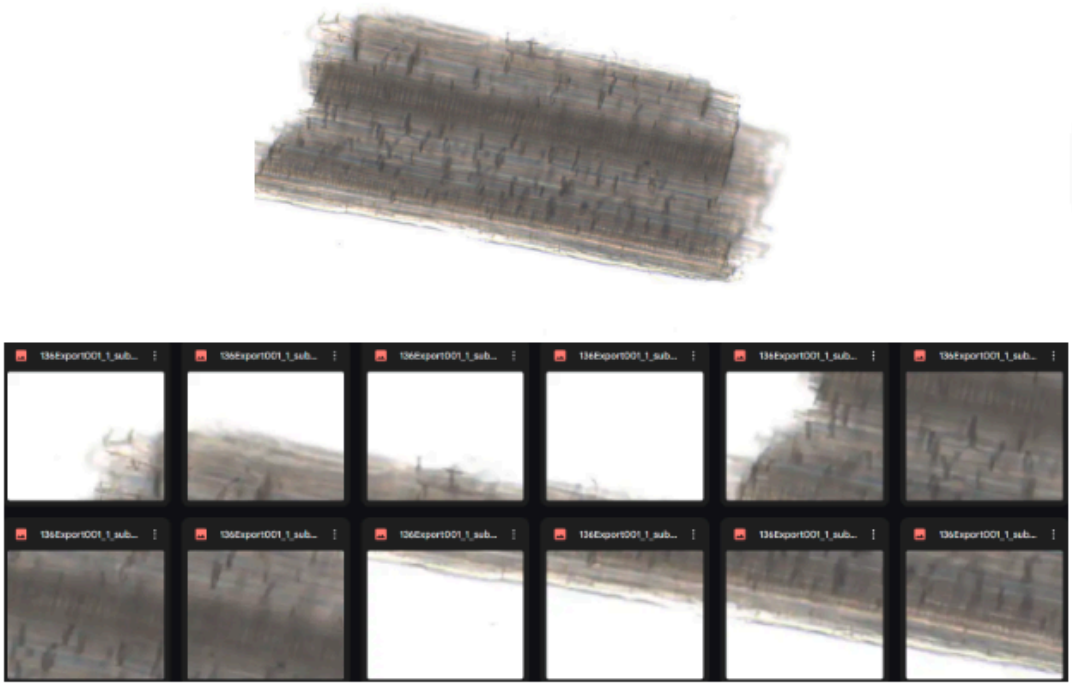

Figure 5: Feature Image to SubImage Splicing

16.3. Keras CNN Architecture

16.3.1. Convolutional 2D Layers

16.3.1.1. Filter Sizes

To accurately and efficiently extract the features from the subspliced microscopic grass training set, two convolutional 2D layers and two max pooling layers all with varying filters are employed in succession with one another. Utilizing different kernel filters on the input images allowed the model to further capture detailed spatial patterns within each subsplice. To accomplish this, first a convolutional layer with a 5x5 kernel size and a 32 filter size is used to process the first set of input images the model would handle. Then A secondary convolutional 2d layer is applied to further capture even finer details for later processing using a 3x3 kernel size paired with a 64 filter size instead.

In between each convolutional layer, a max pooling layer with a 2x2 pool size was implemented to further extract low level edge and texture features before moving the data on to the next layer. The

effect of this combination of convolutional layers and max pooling layers was an increase in the ability for the model to detect hierarchical patterns as the processed data moved through the consecutive layers. This allowed the model to take note of things like simple edges, textures, and other features relevant to the classification of different grass cellular structures.

16.3.1.2. Validation

Approving the model's architecture heavily relied on tweaking the available convolutional parameters to fit the cellular structure data available for training. This was critical in determining how the model's visual parameters would handle generalizing to new and unseen data. To do so multiple layers, filters, and pooling combinations were tested and model convolutional iterations that performed well overall were further validated to reach an architecture consensus that would successfully learn relevant features while being robust across large subsets of the data. In conclusion, the research showed that 2 convolutional layers and 2 max pooling layers was the most effective architecture for the microscopic grass training data.

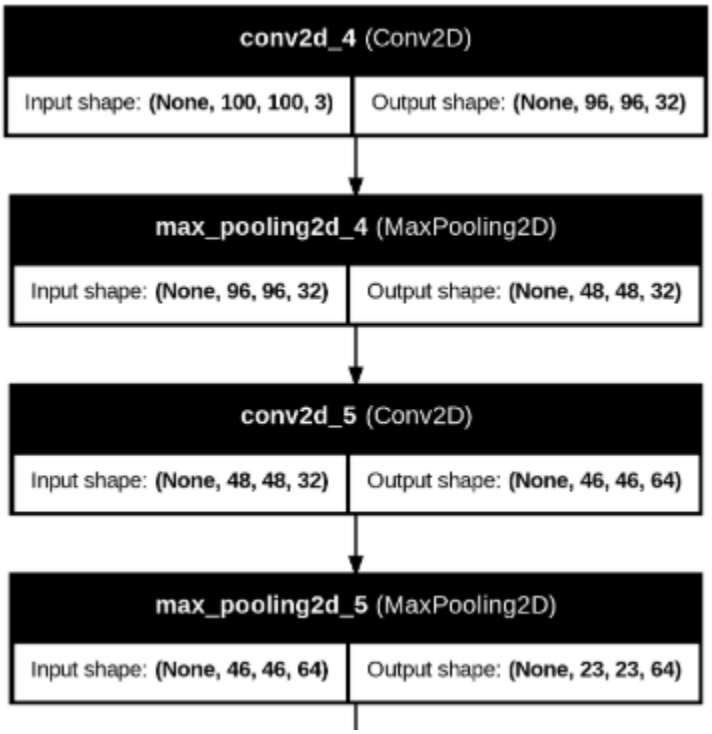

Figure 6: Convolutional 2D Architecture

16.3.2. Dense Layers

16.3.2.1. Complexity Analysis

Once the convolutional layers had generated sets of feature maps, the second half of the model's development required implementation of fully connected dense layers to further interpret those feature maps. First, two dense layers each with a 2128 neuron size were consecutively paired to allow the model to reach its maximum comprehension fastest while conserving its training speed and demand on resources. These layers primarily handle the final decisions the model makes based on its

assigned importance to extracted features and a rectified linear unit or ReLU function is introduced to each dense layer to add an element of nonlinearity to the model.

After the dual dense layers had appropriately analyzed the convolutional feature maps, a final dense layer with four neurons is paired with a sigmoid activation to essentially output a classification score for each of four unique classes of grass cellular structure. In order to stabilize the model and allow it to further avoid overfitting, specific neurons are randomly deactivated during training through a biologically inspired function called Dropout. Therefore both the convolutional and dense layers work in tandem to increase the models ability to capture and interpret complex relationships to its maximum

16.3.2.2. Validation

Validating the fully connected layers depended on finding a right neuron size/activation method that would decrease the computationally expensive process of analyzing all the previous feature maps. As a large number of weights are employed to connect the feature maps to the neuron units, in order to estimate what the right dense layer combinations looks like for a given model it was necessary to repetitively test all the available combinations. From dual dense layers to triple dense layers or differing neuron styles like increasing the neuron density in subsequent layers or working backwards and decreasing it instead. All to ensure that the dense layers don't over complicate the model's solution and that its predictions are effectively representing the validation data.

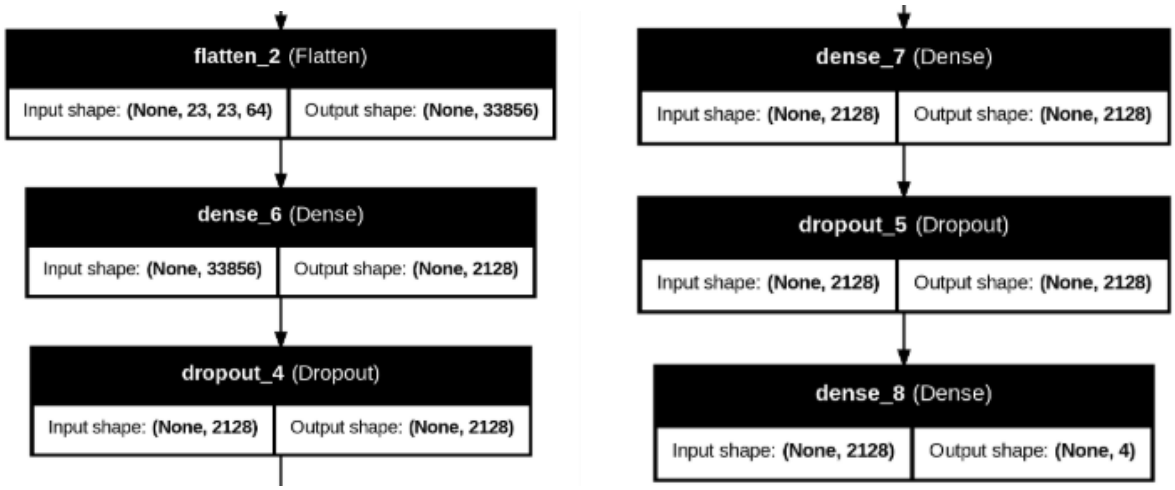

Figure 7: Fully Connected Dense Layers

16.4. Four Feature Percentage Mapping

16.4.1. Optimizer/Loss Functions

The choice of activation, optimizer, and loss functions are the biggest element in introducing true non linearity into the neural network. A non linear network is the key to ensuring that any given output of the model cannot be recreated simply with a specific series of inputs. In order to first introduce non linearity in the GrazeView model, an adaptive learning rate algorithm or Adam was chosen for the optimizer. By using the Adam optimizer the model is capable of providing continuous estimates for the true feature and can dynamically adjust the model's learning rates for any of its individual parameters. Then for the raw probability outputs, the sigmoid activation function was chosen in the

last fully connected dense layer to allow for the model to predict a range of probabilities for all four features for any given input image. Lastly, binary cross entropy was selected for the GrazeView loss function as it is suitable for multi classification problems where there are more than two feature labels.

16.4.2. Validation

To validate the choice of optimizer, activation, and loss functions, comprehensive loss function graphs were generated for each training run of the GrazeView model. If a training run showed a consistent initial decrease of the loss function, it was an indicator that the model was using the optimal network weights to reduce the total classification errors. For example, studying the loss metrics showed that on average for all runs the loss decreased 30 - 40% within the first 20 epochs, demonstrating that our approach with the convolutional neural network was yielding efficient results. This ensured that the network was effectively learning to correctly classify each cellular grass feature.

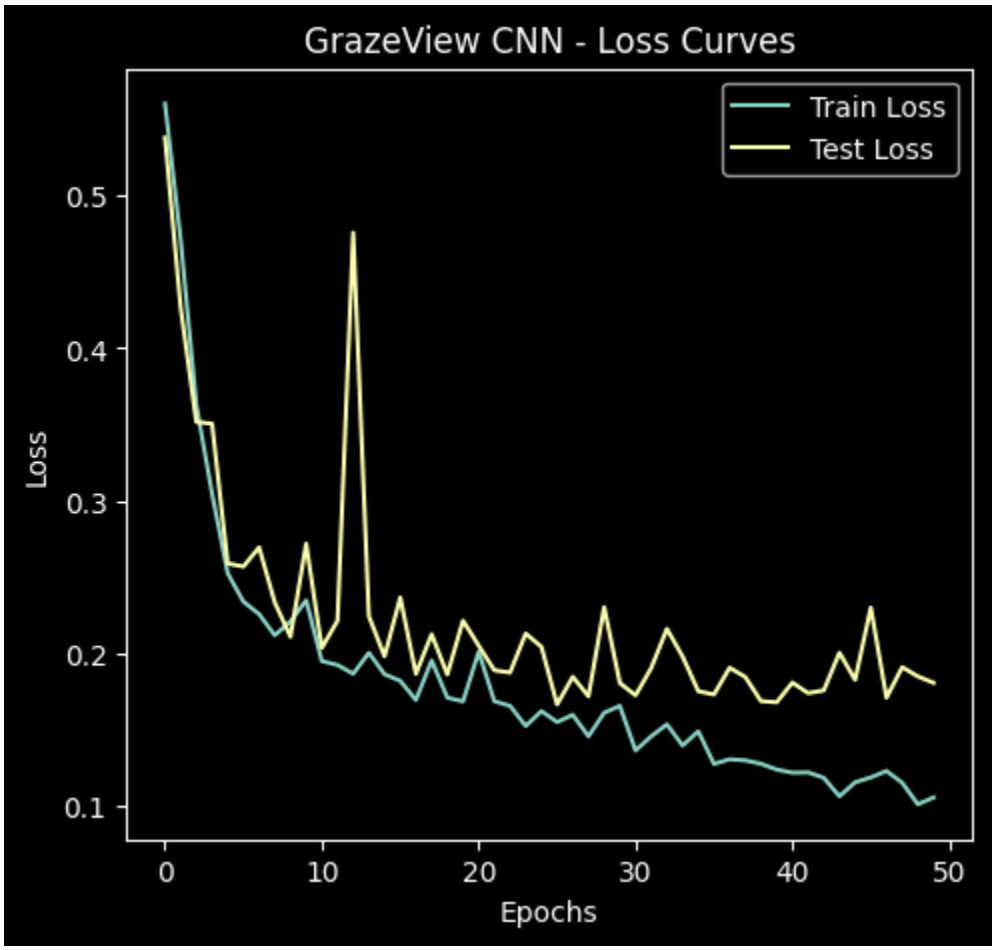

Figure 8: GrazeView Model Loss

16.5. GrazeView Model Training

16.5.1. Network Accuracy

16.5.1.1. Epochs Testing

Ascertaining the network's architecture, optimizers, and loss parameters was the first step in fitting the convolutional neural network to the four cellular sub features. In order to allow the model to learn and refine its weights over time, it was necessary to allow the model to make complete passes over the entire training dataset and thus continuously improve its accuracy over these epochs. To create a robust model, it was essential to have a balanced number of epochs as too little epochs would lead to underfitting where the model is unprepared to handle the initial problem but too many epochs would lead to overfitting where the model memorizes the training data but fails on new input grass images.

16.5.1.2. Validation

Using a trial and error method, 60 epochs was found to be the most appropriate number of epochs to completely classify the training data while avoiding overfitting. By monitoring the validation accuracy of the model, the model effective epochs number could be found by identifying the point in which the model reaches optimal performance while plateauing at the highest accuracy possible. Pairing the overall accuracy metrics with the model's loss curves could then show whether the model was truly learning from the training data or not. In order to further validate the chosen epochs, early stopping was implemented through the model's training to functionally cut off the model from running more ineffective epochs once the loss function showed an upward trend. This not only helped to prevent any unnecessary computation but it allowed the final model to reach a global accuracy of 85%.

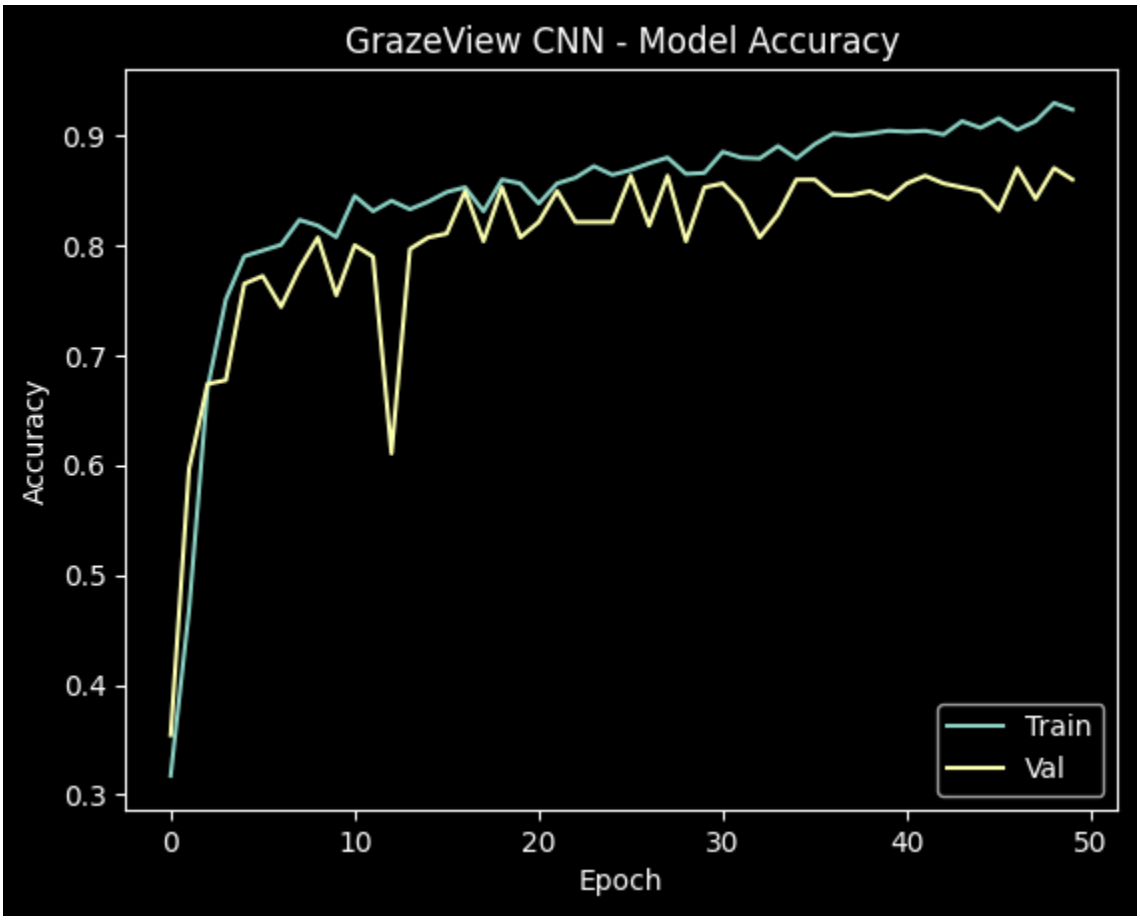

Figure 9: GrazeView Model Accuracy

16.5.2. Feature Verification

16.5.2.1. Individual Metrics

To get a comprehensive evaluation of the model's performance, its output on individual feature metrics was studied to provide insights on how to tweak and improve its training methodology. The four major unique features, nale grass, qufu grass, erci grass, and air bubble were all analyzed to see their individual performance as well as effect on the overall model. To do so, the training dataset was altered in intervals to create imbalanced feature sets that exposed whether the model was overfitting or underfitting on any given feature. Then using these insights, a balanced dataset was applied and the individual accuracy of each feature was plotted. This was repeated across multiple architecture tests, epochs trials, and parameter optimizations to categorize and handle any difficulties the model had with any given feature

16.5.2.2. Validation

Making and validating adjustments to the model's training process heavily relied on the insights made from the individual feature metrics. There were crucial details spotted from these metrics such as the model's similarity of understanding between qufu grass/erci grass due to their comparable cell structure or how the model showed vastly superior performance on the air bubbles due to its visual difference in image properties compared to the other features. For example, air bubbles have a defined circular shape with a starkly contrasting black interior to the white space background which allows the model to correctly classify air bubbles with 95% accuracy across several epochs and architecture styles. All these independent insights allowed for the individual feature validation to avoid overfits towards any single class and instead make accurate predictions for all feature classes.

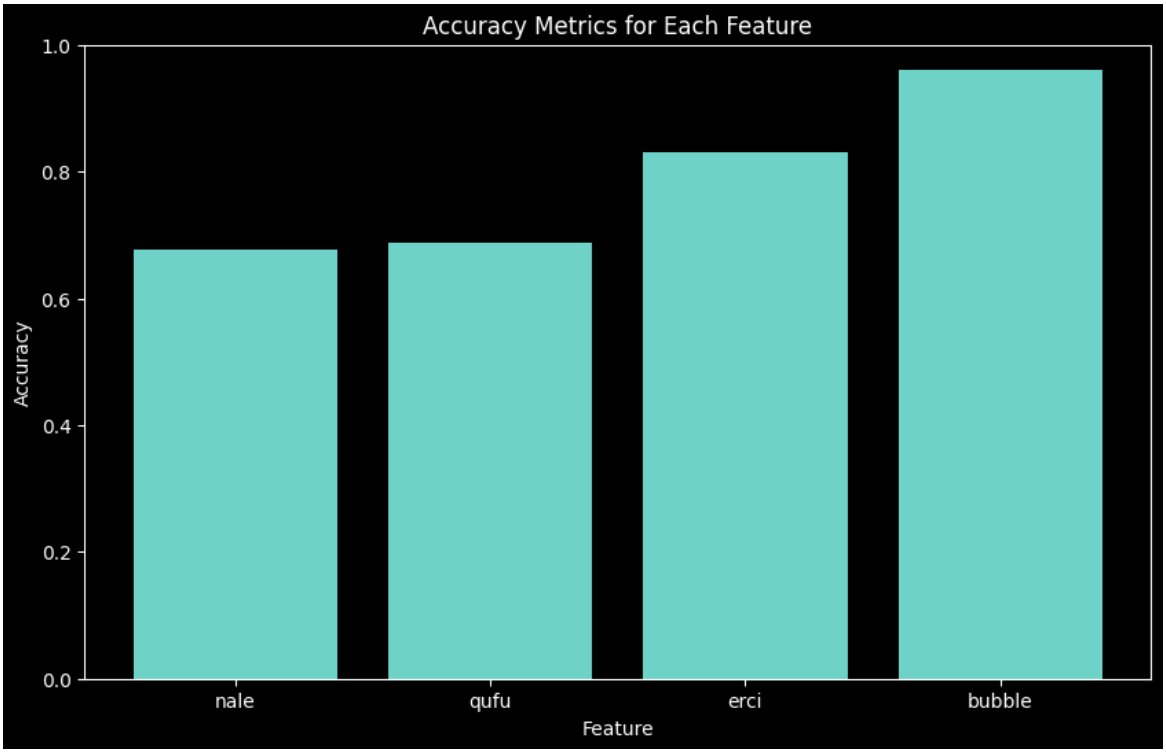

Figure 10: Individual Grass Feature Accuracy

16.6. New Image Processing

16.6.1. Multilabel Classification

Newly inputted microscope grass images follow the same format as the sub image processing methodology. Each 20000 x 20000 pixel microscope image is split into hundreds of mini images each 100 x 100 pixel size all prepared and ready for analysis by the trained model. As each image can contain more than one feature, the neural network has already been constructed to handle multilabel-multiclass classification through the combination of the sigmoid activation to create a range of predicted probabilities and the binary cross entropy loss function to handle the independent decisions necessary for each class. Each new sub image fed into the model will output a vector series of four percentages each corresponding to nale grass, qufu grass, erci grass, and air bubbles that all sum up to one. Using all the processed sub images, their unique percentages will be added up to a final percentage count of the entire image to show what ratio of features is present in each microscopic grass slide. This ensures that the final output predictions will be insured to handle real world agricultural applications where the cells on the microscope slides exhibit mixed characteristics.

16.6.2. Validation

To ensure a seamless usage flow for the client facing aspect of the GrazeView Model plus the GrazeView Web Application, the total time for a new image to be uploaded through the virtual software and processed by the GrazeView Model has been minimized to 45 minutes. This allows for users to process multiple cell slides in single sessions more accurately and efficiently than through a manual rigorous process. As the model also displays over 80% accuracy on accurately categorizing the unique features, GrazeView users can have confidence in the credibility of the network's predictions.

Figure 11: Sample New Image Classification

16.7. Future Development

16.7.1. Integration with User Interface

The GrazeView Model outputs will replace the current placeholder values in the GrazeView Web Application. The trained GrazeView Model will be backend connected to the web application through a FastAPI framework which will ensure fast and accurate real time data processing through routine HTTPs requests. The ML model will then assist in providing reliable labels and feature predictions that will be dynamically displayed on the User Interface.

16.7.2. Integration with Database

The GrazeView Model will intercept CSV's created from the GrazeView Database and use the processed splices to categorize a percentage map of found features. The bounding box entries and unique IDs will be used to classify user inputs which will drive network predictions, metadata, and any user information to be stored on the User Interface Dataviewer. The GrazeView Database will encapsulate the new data flow automation to help assist in model retraining.

16.7.3. Deployment

The current GrazeView Model is publicly hosted on Google Colab and executed using the Tesla T4 GPU on the Google Turing Architecture. Future development/deployment will be moved to a scalable cloud platform such as Amazon Web Services to more efficiently handle the API requests.

16.8. Subsystem Conclusion

The final trained GrazeView Model demonstrated a significant improvement in performance and operation of the past iterations of the AgriLife neural network project due to the new implemented image processing techniques such as the sub image feature splicing, the refined convolutional/dense architecture, and the optimizations of the model's multi labeling capabilities through adaptive rate learning and the binary cross entropy loss function. Furthermore, the model successfully achieved a global accuracy of over 85% and a minimum accuracy of at least 70% for all features. Additionally, the model displayed an impressive strength in identifying features such as air bubbles, boasting close to 99% accuracy when it came to identifying unique edge variations. Pairing this with the streamlined data, the model displays an impressive ability to quickly and accurately characterize the necessary cellular features in under 45 minutes. This allows for the GrazeView Model to be highly practical for use in rural farm work, agricultural research, and wide scale crop management.

17. U.I. Functionality Subsystem - Jack Nelson

17.1. Subsystem Introduction

The GrazeView Application is the subsystem which serves as the interface between the user and the Machine Learning Model. For this subsystem, a front-end GUI is built using the WinForms Framework, which is a part of the Microsoft .NET platform. Winforms is a robust and versatile framework for building desktop applications with an emphasis on user-friendly design and rapid development.

The application allows users to upload data via a dedicated upload form, initiating a process that transitions them to a loading page where the system processes the input. Once processing is complete, users are taken to a results page that displays both the user-provided input and the results generated by machine learning algorithms. This seamless flow ensures users can efficiently upload, view, and interpret their data.

Additionally, the application includes a data library, where users can browse all previous uploads. From this library, users can select specific uploads to review or print, providing a convenient and organized way to manage their data. By combining user-friendly navigation with robust data handling capabilities, the WinForms interface ensures an accessible and efficient experience for all users.

17.2. U.I. Architecture

The user interface architecture of the GrazeView Application is designed using the Windows Forms framework, which supports modular development and provides a seamless connection between the front-end and back-end logic. The architecture follows a structured approach to ensure scalability, maintainability, and ease of user, as pictured in *Figure 6*. The pathflow of how the user can navigate the application is also shown in *Figure 6.*

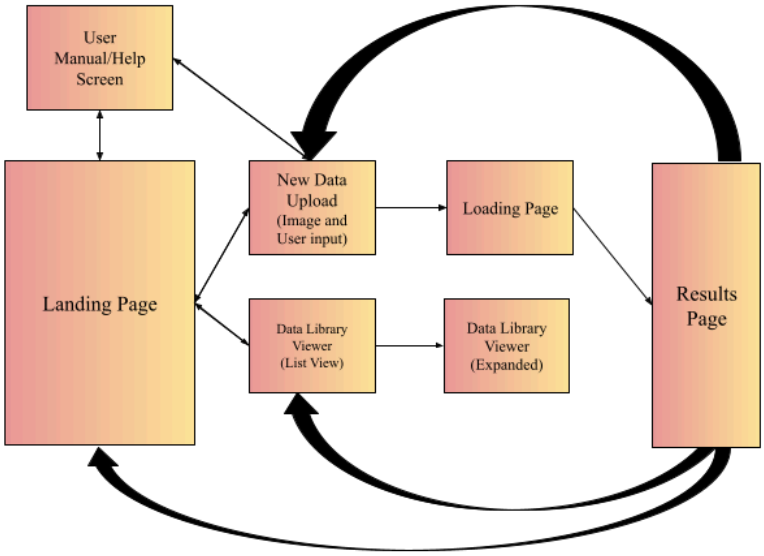

Figure 12a : U.I. Pathflow

Figure 12b : U.I. Pathflow with Screenshots

17.2.1. Component-Based Structure

The user interface is divided into several key components, each responsible for a specific aspect of the application. These components include:

17.2.1.1. Upload Form

The upload form is designed for data input, as well as file uploads. This form captures user inputs, such as sample time and location, and ensures that these fields are validated prior to submission. The full list of user input fields are listed as:

- Upload Name (Optional)
- Sample Location(Optional)
- Sample Time/Data (Required)
- Sheep Breed(Optional)
- Comments(Optional)

Figure 13 : Upload Form

17.2.1.2. Loading Form

A transitional interface is used to visually show the progress of the data processing. This page ensures a responsive user experience by keeping the user informed while backend operations are underway.

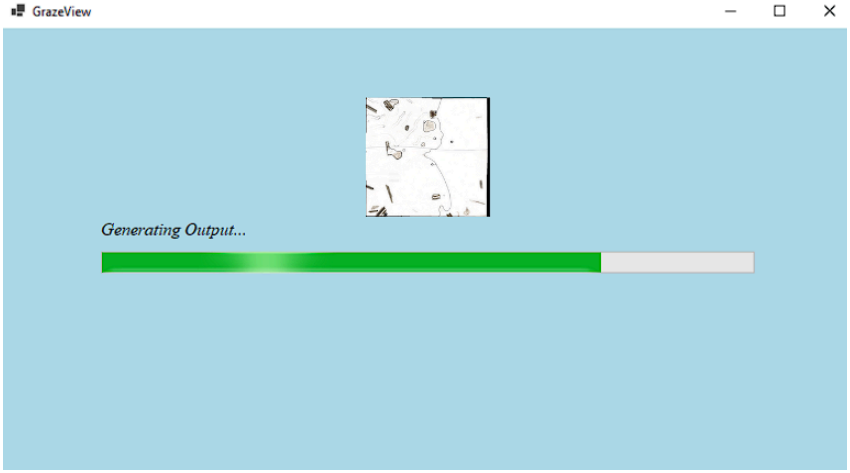

Figure 14 : Loading Form

17.2.1.3. Results Form

The results form provides a display interface that presents both the user's provided data and the results from the machine learning model. This page provides clarity, as well as well-organized data presentation.

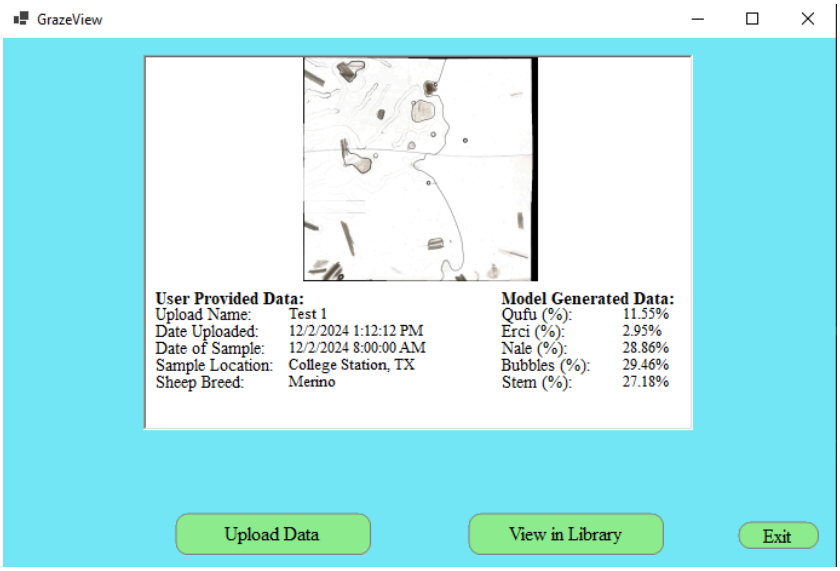

Figure 15 : Results Form

17.2.1.4. Data Library

The data library provides a centralized interface for viewing, selecting, and printing all previous uploads. This form supports search, filter, and clearing functionality for user convenience.

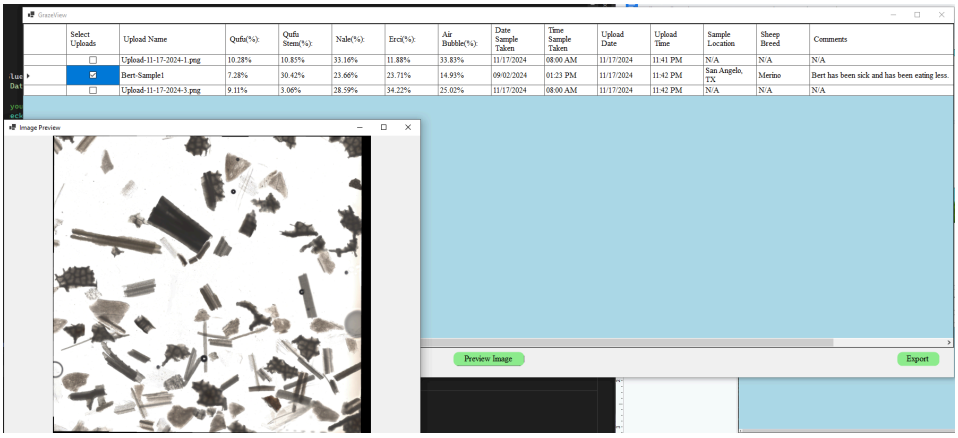

Figure 16 : Data Library

17.2.1.5. User Guide

The user guide form provides access to help while navigating and using the GrazeView Application. It is available on most forms, through the help icon in the upper right corner, and provides step-by-step instructions on how to upload, view, print, and clear data within the application.

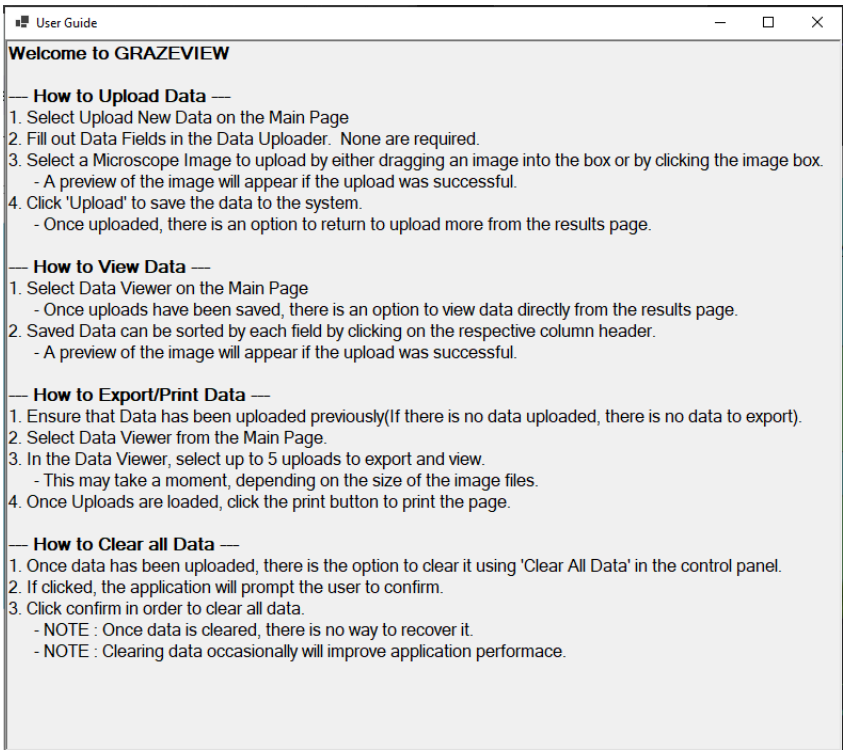

Figure 17 : User Guide

17.2.2. Event-Driven Design

The GrazeView Application's user interface is built using an event-driven programming model, which forms the backbone of its functionality. Each user interaction—such as clicking a button, entering data, or selecting an item—triggers specific events that are processed by the application. For instance, the "Upload" button on the Upload Form validates the required fields and initiates a transition to the Loading Page, where background operations handle data processing. Similarly, selection events in the Data Library enable users to interact with their uploaded data, allowing them to choose specific entries for review or printing. Validation events ensure that all inputs meet predefined criteria before advancing to subsequent stages, thereby reducing errors and improving the overall user experience. This event-driven approach ensures that the application responds dynamically to user actions, making it both intuitive and efficient.

17.3. Validation Plan

17.3.1. U.I. Component Validation

Each feature within the application undergoes several verification tests. If the application is able to pass validation, they are considered “Complete.” All validation tests are shown below in *Table 6.*

Feature	Test	Status
U.I.	Main Page Resize Handled	Complete
	Data Upload Resize Handled	Complete
	Loading Page does not resize	Complete
	Results Page Resize Handled	Complete
	Data Library Resize Handled	Complete
	DL Expanded Resize Handled	Complete
	Pages remain consistent in sizing when navigating from one form to another	Complete
	All buttons that take the user to another form perform correctly. This pathflow follows *Figure 6.*	Complete
Data Upload	Data can be uploaded by the user and is stored within the data library. Uploads are correctly being stored separately from one another.	Complete
Loading Page	The loading page correctly gives the user real-time updates on the ML’s progress. NOTE - This is currently on a timer. This page will be adjusted in ECEN 404 when the ML is connected.	Complete
Results Page	The results page and data library are correctly passed data from the Machine Learning model. These results are able to be	Complete

	viewed. NOTE - Currently, ML Data is being generated by a random number generator. This will be adjusted in ECEN 404.	
User Guide	Necessary pages have access to the user guide, through the 'help icon.' Only one instance of the user guide can be opened at one time.	Complete
Data Storage	Uploads are able to be saved and retrieved when the application is closed. NOTE - This will be adjusted when the Database is connected in ECEN 404.	Complete
	The Data Library is able to be completely cleared via "Clear all Data" button. This function also has validation to give the user a chance to change their mind.	Complete
	The Data Library is able to be sorted by different data fields.	Complete
DL Expanded	The data library allows the user to select multiple uploads to view all data and the image in a print-format.	Complete
	The DL Expanded View page allows the user to print the page, only capturing important features.	Complete

Table 6 : U.I. Functionality Validation

17.4. Subsystem Conclusion

The GrazeView Application's user interface (U.I.) subsystem successfully meets all validation criteria, ensuring a reliable, efficient, and user-friendly experience. The validation tests demonstrated that the application handles resizing across all forms, maintains consistent page transitions, and correctly processes data uploads, library interactions, and results visualization. All user-facing functionalities, such as uploading data, navigating forms, and managing previous uploads, have been implemented and tested to operate as intended.

The U.I. subsystem is designed to provide a seamless interaction between the user and the backend processes, including real-time data processing and machine learning integration. Although some features, such as machine learning outputs, currently rely on placeholders (random number generation), these will be refined in subsequent development phases (e.g., ECEN 404). The successful completion of component validations reflects the robustness and scalability of the subsystem, ensuring it can accommodate future enhancements with minimal disruption.

With its modular design and validated features, the U.I. subsystem provides a solid foundation for the GrazeView Application. It ensures accessibility, consistency, and functionality, meeting the requirements for efficient data management and visualization. As further enhancements and integrations are implemented, the subsystem is well-positioned to continue delivering a high-quality user experience.

17.5. Future Development

Although the current GrazeView Application's U.I. subsystems meet all functional and validation requirements, there are several areas for future development to enhance the system's scalability, usability, and integration capabilities. These improvements will ensure the application can support broader use cases and a larger user base in the future.

17.5.1. Integration with Database Subsystem

Currently, data storage and retrieval are implemented as standalone operations. Future development will involve connecting the system to a robust database, enabling persistent data storage and advanced query functionalities. This will streamline the management of uploads and allow for more efficient sorting, filtering, and searching within the Data Library.

17.5.2. Machine Learning Integration

At present, machine learning (ML) outputs are generated using placeholder methods, such as a random number generator. In future updates, the ML algorithms will be fully integrated, providing accurate and meaningful results based on the uploaded data. This integration will involve connecting the backend to trained ML models and ensuring real-time data processing.

17.5.3. Scalability and Deployment

The current version of the application operates in a local environment. To support a larger user base, future development will focus on deploying the application in a scalable environment using cloud platforms such as Microsoft Azure or Amazon Web Services (AWS). This deployment will ensure better performance, availability, and security.

17.5.4. Security Enhancements

As the application evolves, it will no longer be a local application and will have access to cloud platforms, such as Microsoft Azure or AWS. With this, there will be a greater need to include personal logins, data encryption, and SSL certificates. These features will allow the application to be used more widespread, while still allowing users to protect the data they are uploading.

17.6. Software Requirements and Documentation

The GrazeView Application is designed to function as a standalone desktop application, leveraging a robust software environment and providing comprehensive documentation to ensure usability, maintainability, and troubleshooting support.

17.6.1. Software Requirements

The application is built to operate efficiently within the following minimum system requirements:

17.6.1.1. Operating System

This application requires Windows OS, ideally Windows 10 or later, in order to ensure compatibility with the .NET Framework and Windows Forms platform.

17.6.1.2. Memory(RAM)

A minimum of 10 GB of RAM is required to handle the processing of large image files and machine learning data efficiently.

17.6.1.3. Storage

At least 10 GB of available disk space for storing application data, user uploads, and temporary files.

17.6.1.4. Processor

At least 10 GB of available disk space for storing application data, user uploads, and temporary files.

17.6.2. Documentation

The GrazeView Application includes detailed documentation to support its users and developers. The documentation is designed to ensure proper operation, troubleshooting, and future maintainability:

17.6.2.1. User Guide

A comprehensive guide is integrated into the application and accessible via a "Help" icon. It includes step-by-step instructions for all core functionalities, such as uploading data, navigating forms, and managing the Data Library. Only one instance of the guide can be opened at a time to avoid confusion.

17.6.2.2. External Help Sources

This application uses third-party external packages, and lists the dependencies and links to those sources, in order to help further development. These packages are as follows:

Document	Revision/Release Date	Publisher
Desktop Guide(Windows Forms .NET)	.NET 8.0	Microsoft
TensorFlow Core	2.0	Google

Table 7 : Reference Documents

17.6.2.3. Developer Documentation

For future development and maintenance, the application includes internal documentation of its methods, algorithms, and architectural design. This documentation highlights the use of third-party external packages, their dependencies, and potential areas for customization or extension.

18. U.I. Visual Design Subsystem - Nathan Scott

18.1. Subsystem Introduction

The U.I. Visual Design Subsystem is a crucial component of the GrazeView Application, responsible for the graphical presentation and overall user experience. This subsystem focuses on creating an intuitive and visually appealing interface, ensuring seamless navigation and ease of use for all users.

The design of the user interface was developed using the WinForms Framework, leveraging its capabilities to implement consistent and interactive visual elements. The subsystem focuses on the placement and sizing of interactable visual elements (buttons, labels, textboxes, data tables, progress bars, etc.), form properties, and background designs. All pages in the GrazeView Application should have consistent graphical components and properties for the user.

Good visual design is essential in any application as it serves as the bridge between users and the underlying functionality. A visually appealing and well-organized interface not only enhances the overall user experience but also ensures clarity and accessibility for a diverse audience. It should help guide users in a way that reduces errors, confusion, and frustration. The visual design should reinforce the confidence and reliability of the application. The subsystem plays a vital role in presenting the information and data that the user provides in a digestible and visually coherent manner. These are all important elements of this subsystem that were taken into consideration when implementing the design in the application.

18.2. Individual Page Designs

A total of seven pages were designed. The pages include: Start page, Data Upload page, Data Library page, Loading page, Results Page, User Manual page, and the Expanded View page. The following sections will cover the visual components of each of these pages along with their purpose.

18.2.1. Landing (Start) Page

This page serves as the central hub for users. This will be the first page users will see when working on the application, so it is important to introduce a visual theme and style for the remaining pages of the application.

Figure 18: Landing (Start) Page

18.2.2. Data Upload Page

This page is available to the user as a way to input their data about the samples they wish to be analyzed. The user can use the right box to upload their sample image file and proceed to write specific information regarding the sample on the left.

Figure 19: Data Upload Page

18.2.3. Data Library Page

This page is available to the user as a way to be able to view all of their uploaded samples, data, and information in an efficient way. Users will be able to select uploads, sort the data, preview the image for each upload, and export as a printable .pdf file if they choose. All data inputted by the user from the "Data Upload" page about the sample they uploaded will show up on this page. The user may also choose to clear all data from the table if they wish.

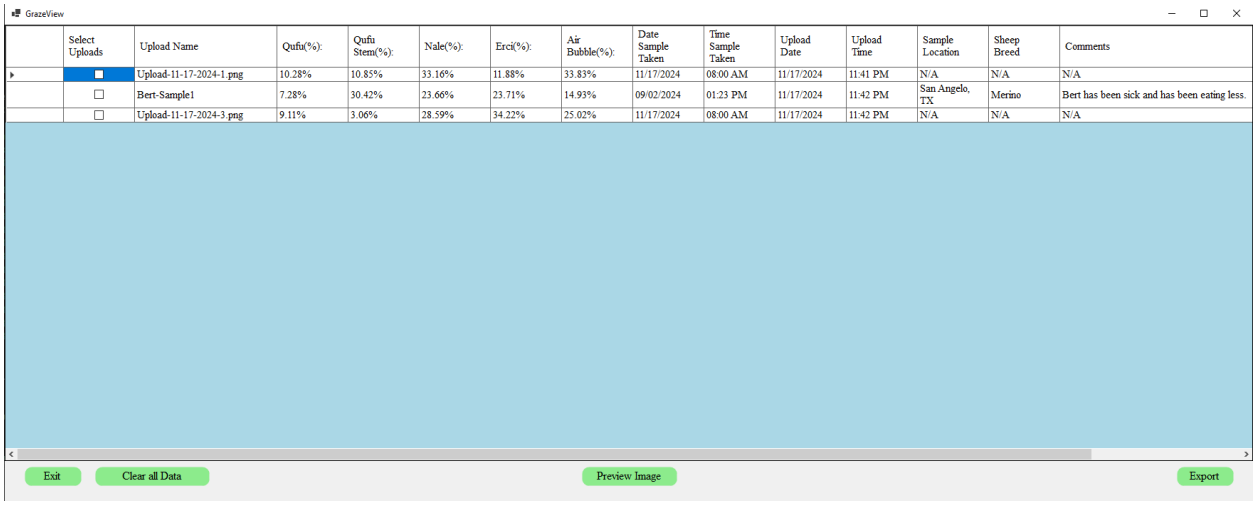

Figure 20: Data Library Page

18.2.4. Loading Page

This page acts as a temporary buffer between the "Data Upload" page and the "Results" page until the Machine Learning Subsystem is connected.

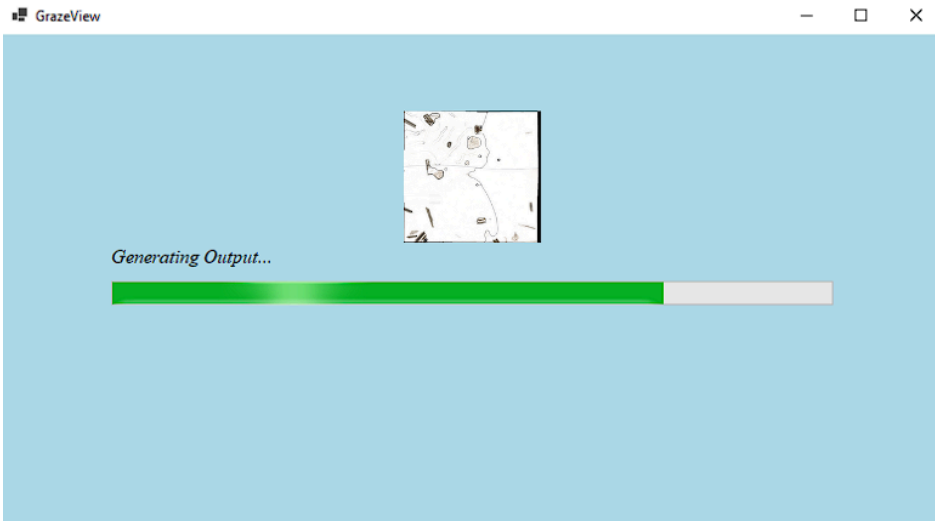

Figure 21: Loading Page

18.2.5. Results Page

This page will display the data produced from the Machine Learning Subsystem concerning the uploaded image file. The data produced will be the percentages of the types of grass analyzed in the image. This page will also reproduce any user-inputted data from the "Data Upload" page. From here the user can upload more data (Data Upload page), view the data library (Data Library page), or return again to the "Landing" page.

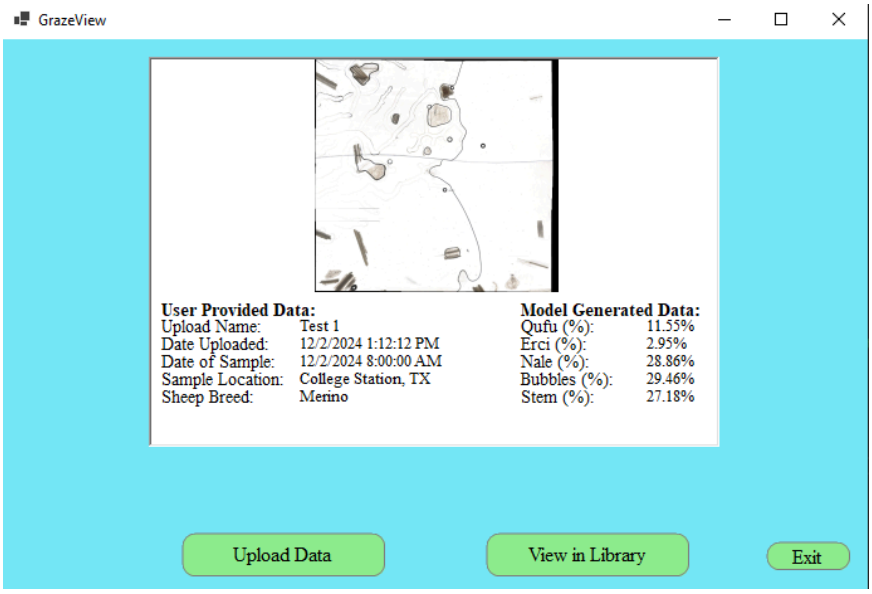

Figure 22: Results Page

18.2.6. User Manual Page

This page will open if the user clicks on the question mark in the top right corner of the “Landing” page or the “Data Upload” page. This page serves to instruct the user on any functionalities of the application they may be concerned with. Only one instance of this page can be open at a given time.

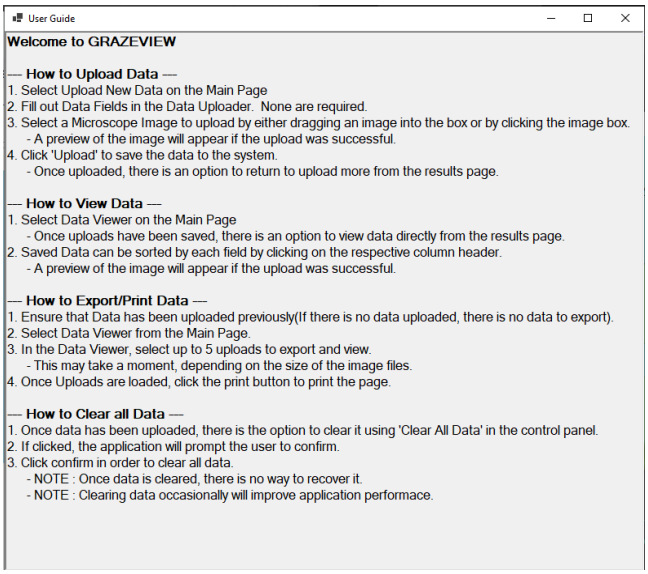

Figure 23: User Manual Page

18.2.7. Expanded View Page

This page will open if the user presses the “Export” button in the “Data Library” page. This page will present all selected uploads along with their individual data and image files for the user to analyze. The user may print a .pdf file of the image from this page if they wish to do so.

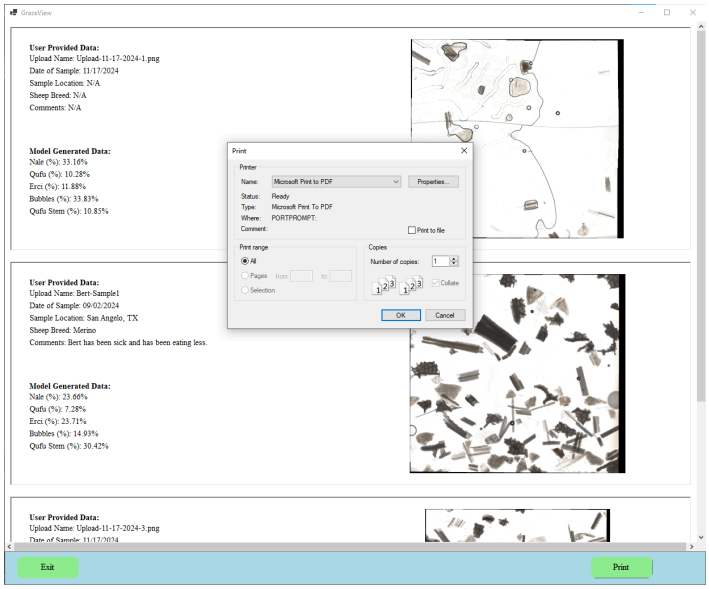

Figure 24: Expanded View Page

18.3. Validation Plan

18.3.1. U.I. Visual Design Validation

Validation for this subsystem was done similarly to that of the U.I. Functionality Subsystem. The following table shows the methodology for the validation of this subsystem.

Feature	Test	Status
U.I.	Implemented Background Design	Complete
	Consistent Properties across All Pages	Complete
	New Rounded Button Component Operates Correctly	Complete
	All Components Placed and Sized appropriately	Complete
	All Design Components stay consistent across pages	Complete
	All Design Files are embedded within the entire Project	Complete
Results Page	Data is presented in an efficient layout	Complete
Data Library Page	Data is being gathered in the table after uploading	Complete
	Preview Window displays the uploaded image	Complete

Table 8: U.I. Visual Design Validation

18.4. Subsystem Conclusion

The U.I. Visual Design Subsystem successfully achieves its goal of providing a visually cohesive, intuitive, and user-friendly interface for the GrazeView Application. The visual design effectively complements the application's functionality, guiding users through data upload, processing, and results interpretation with ease. The designed pages focused on aesthetics and usability, ensuring an engaging and efficient user experience. The consistent use of graphical highlights and modern UI features demonstrates a thoughtful approach to enhancing navigation and interaction. Validation tests confirm that the visual elements perform as intended, maintaining compatibility with the application's resizing, transitions, and overall structure. This subsystem opens the door for future development on the application.

18.5. Future Development

Although the current UI Visual Design Subsystem effectively meets all functional and validation requirements, there are several opportunities for future development to further enhance usability, accessibility, and visual interactivity. These improvements will ensure the interface remains modern, intuitive, and capable of supporting the application's evolving functionality.

18.5.1. Interactive Image Editing

A significant area for future enhancement is the development of tools to edit user-uploaded images directly within the application. This will include visual methods to highlight and differentiate specific regions, such as sections of grass or air bubbles, using overlays, color-coded markers, or dynamic bounding boxes. These features will improve the interpretability of analysis results and allow users to better understand the system's processing of the images.

18.5.2. Advanced Visualization

Future iterations could also introduce interactive visualizations, such as zoomable or pan-able image sections and layered overlays that dynamically update to reflect machine learning predictions. These improvements would make it easier for users to engage with and explore the visual data.

18.5.3. Enhanced Accessibility Features

To broaden usability, the subsystem could implement accessibility-focused enhancements, such as adjustable font sizes, high-contrast modes, and customizable color schemes. These changes will cater to a more diverse user base and align the application with modern accessibility standards.

18.5.4. Integration with Real-Time Processing

As machine learning integration becomes fully realized, the UI Visual Design Subsystem will need to support real-time updates and visualizations. This will involve creating responsive designs that display results dynamically, ensuring that users receive immediate feedback in a visually clear and structured manner.

By focusing on these future developments, the UI Visual Design Subsystem will continue to deliver a high-quality user experience while keeping pace with advancements in functionality and technology. These enhancements will further cement GrazeView AI as a professional and accessible tool for microhistological analysis.

19. Database and Image Processing Subsystem - Arnav Gokhale

19.1 Subsystem Overview

The Database and Image Processing Subsystem integrates advanced image splicing techniques with a robust dual-database architecture to preprocess large microscopic images and manage feature data efficiently. It ensures:

- High-quality image cleaning and segmentation.
- Precise feature detection using bounding boxes.
- Seamless integration with the machine learning (ML) model for classification.
- Dynamic updates to two databases:
  - The Image Database for feature occurrences and percentage breakdowns.
  - The Feature Database for bounding box coordinates and detailed feature classifications.

This subsystem enhances the accuracy and efficiency of the GrazeView application while offering robust data storage and retrieval capabilities.

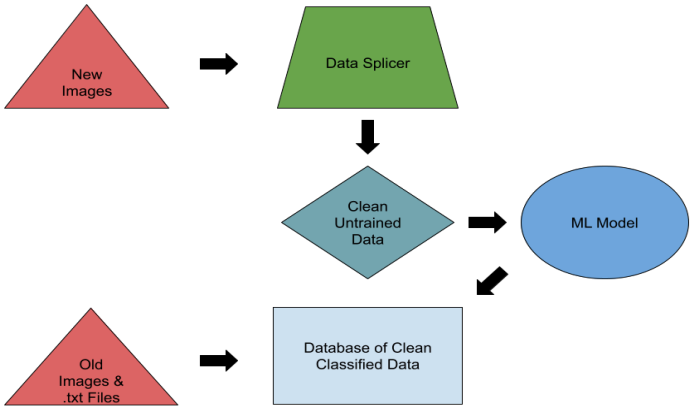

Figure 25: Database/Image Splicer Pipeline Flow

19.2 Image Splicer

19.2.1 Image Cleaning and Preprocessing

- Grayscale Conversion: Simplifies image data, reducing computational load while focusing on structural details.
- Gaussian Noise Filtering: Smooths images to remove high-frequency noise, clarifying feature edges.
- Canny Edge Detection: Identifies edges and boundaries, aiding in feature localization.
- Brightness Normalization: Ensures uniform exposure across images, mitigating variations from different setups.
- Whitespace Removal: Removes regions with whitespace to optimize storage and processing.

- Kernel: Uses a small matrix (kernel) to apply morphological operations. Enhances feature connectivity and smoothness in the image.
- Dilation: Expands the edges of detected features to connect close but distinct features. Ensures features that are near each other are treated as a single entity for bounding box generation.
- Thresholding: Applies binary thresholding to separate regions of interest from the background based on pixel intensity. Creates a clear distinction between features and the surrounding area.
- Bitwise Operation: Combines or excludes specific image regions using bitwise AND, OR, and NOT operations. Refines feature boundaries, ensuring accurate detection and bounding box creation.
- Contours Detection: Extracts the contours (outer boundaries) of detected features in the image. Provides precise localization and shape information for each feature, supporting accurate bounding box generation.

19.2.2 Image Splicing and Feature Detection

- Subimage Creation: Large images are divided into smaller, manageable subimages to focus on localized ROIs.
- Bounding Box Generation:
  - Identifies potential features within subimages.
  - Creates bounding boxes using encoded coordinates:
    - Center: Midpoint of x and y axes.
    - Dimensions: Width and height of the bounding box.
- Integration with ML Model
  - Subimages and bounding boxes are sent to the ML model.
  - The model classifies each feature and sends back numeric encodings:
    - Nale = 0, Qufu = 1, Erci = 2, Air Bubble = 3, Qufu Stem = 4.

19.2.3 Automation

- Automated Workflow
  - Cleans and splices images upon upload.
  - Detects features, generates bounding boxes, and updates databases without manual intervention.
- Real-Time Integration
  - Updates databases dynamically for immediate availability of results.

19.3 Database Organization

19.3.1 Feature Database

- Purpose: Logs detailed feature data, including bounding box coordinates and classifications.
- Key Fields:
  - Index, Feature Type, x-center, y-center, x-length, y-length, Source File.

Example:

Feature Database

Index	Feature Classifi	1	2	3	4	Source_File
0	0	0.042651	0.246544	0.054053	0.077287	105 Export001.txt
1	0	0.79834	0.273389	0.068066	0.054782	105 Export002.txt
2	0	0.294922	0.462578	0.069043	0.054054	105 Export003.txt
3	0	0.47688	0.37474	0.044287	0.047505	110 Export004.txt
4	0	0.733716	0.80907	0.099658	0.077599	111 Export003.txt
5	0	0.776855	0.929834	0.041309	0.084927	112 Export001.txt

Table 9: Feature Database

- Automation:
- Bounding box coordinates and classifications are added dynamically as images are processed.

19.3.2 Image Database
- Purpose: Provides a high-level summary of feature occurrences and percentage breakdowns within each image.
- Key Fields:
- Source File, Feature Counts (Nale, Qufu, etc.), Feature Percentages.

Example:

Image Database

Index	Source File	Nale	Qufu	Erci	AirBubble	QufuStem	Nale %	Qufu %	Erci %	AirBubble %	QufuStem %
0	105 Export001.txt	8	23	3	0	0	15.326438904128700	4.41499129433992	36.30018574785070	8.831544674936880	35.126839378743800
1	105 Export002.txt	9	17	4	0	0	33.78631472285470	1.1169247711280400	27.71507745213520	22.136170153324200	15.245512900557900
2	105 Export003.txt	6	16	1	0	0	25.714182510700800	22.356432932286900	22.553901041862100	2.998167288440260	26.377316226709900
3	110 Export004.txt	7	1	8	0	0	33.19648702378860	17.431498128098000	4.6656839080574200	2.8658268970194200	41.84050404303660
4	111 Export003.txt	6	3	0	0	0	10.681112436281000	34.81699826843120	11.065180822499400	36.37679797249610	7.059910500292290
5	112 Export001.txt	10	6	4	0	0	4.580928941325380	26.651561334490100	26.38690691635510	15.876269691672900	26.504333116156600

Table 10: Image Database

- Automation:
- Automatically adds feature counts and percentages from the processed data to the existing database.

19.4 Integration Workflow

1. Image Upload:
- Users upload images via the application.
- Images are assigned unique identifiers for traceability.

2. Image Processing:
- The Image Splicer cleans and splices the uploaded images.
- Features are detected and bounding boxes generated.

3. Feature Classification:
- Bounding boxes and subimages are sent to the ML model for classification.
- ML model returns classifications, which are encoded numerically.

4. Database Updates:
- Feature Database: Logs bounding box coordinates, classifications, and source file references.
- Image Database: Updates feature counts and calculates percentage breakdowns.

5. User Accessibility:
- Data is made available in real-time for user analysis through the application.

19.5 Benefits

19.5.1 To Users
- High-Level Insights:
  - Summary statistics in the Image Database allow users to quickly identify trends and anomalies.
- Detailed Analysis:
  - Spatially resolved data in the Feature Database supports precise validation and targeted investigations.

19.5.2 To Machine Learning Integration
- Optimized Inputs:
  - Cleaned, spliced images ensure consistent, high-quality inputs for the ML model.
- Scalable Training:
  - Dataset expansion through splicing improves ML robustness and accuracy.

19.5.3 To System Performance
- Automation:
  - Eliminates manual preprocessing and data management, enhancing scalability.
- Efficiency:
  - Processes and analyzes large datasets quickly, minimizing computational overhead.

19.6 Validation and Testing

19.6.1 Image Splicing Validation
- Ensures accuracy of edge detection, bounding box creation, and sub image processing, through image visualizations. Below are some images of the same region of interest, run through different versions of the image splicer code, showing the increased accuracy.

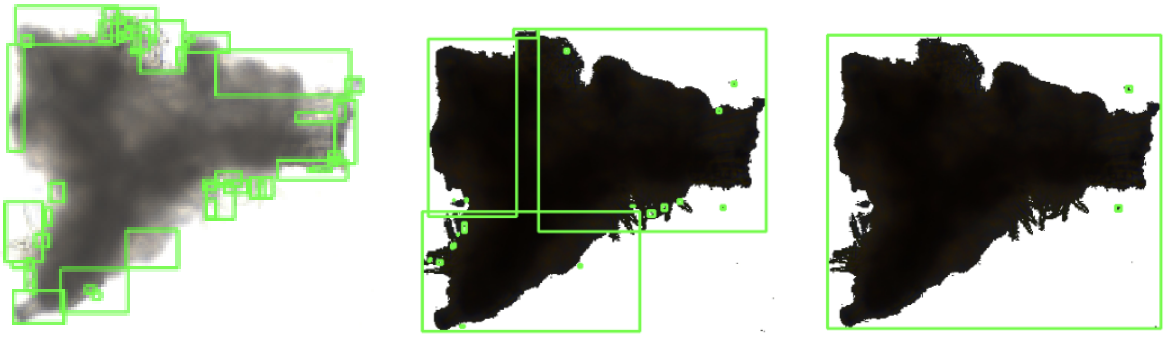

Figure 26: Image Validation (Improved Bounding Box Accuracy)

The Image Splicing subsystem underwent multiple iterations to improve its accuracy in edge detection, bounding box creation, and sub image processing. Each version incorporated feedback and testing results, leading to enhanced performance and reliability. Validation was performed using image visualizations that demonstrated the progressive improvements in detecting and isolating features. Below is a summary of the different versions and how they evolved.

Initial Version: Basic Functionality
Features:
- Implemented simple grayscale conversion and basic thresholding to isolate regions of interest.
- Generated bounding boxes based on pixel intensity without morphological operations.

- Limitations:
- Over-segmentation occurred, with multiple small bounding boxes for features that were close together.
- Missed faint or partially connected features due to inadequate edge detection.

- Validation:
- Images showed disconnected edges and fragmented bounding boxes around single features.

Version 2: Improved Edge Detection
- Enhancements:
- Introduced Canny edge detection for more precise boundary identification.
- Added Gaussian noise filtering to smooth images and reduce unnecessary noise.

- Improvements:
- Reduced over-segmentation by focusing on meaningful edges.
- Detected finer features that were previously missed.

- Validation:
- Visualizations showed more cohesive bounding boxes, but some features remained fragmented.

Version 3: Morphological Operations
- Enhancements:
- Introduced kernel-based morphological operations (dilation and erosion).
- Dilation connected nearby edges, ensuring close features were treated as a single entity.

- Improvements:
- Significantly reduced fragmented bounding boxes.

- Improved handling of features with intricate or faint edges.

- Validation:

- Images demonstrated well-connected and accurately bounded features, though some noise persisted.

Version 4: Advanced Preprocessing

- Enhancements:

- Added brightness normalization to handle variations in image exposure.
- Applied bitwise operations to refine regions of interest after thresholding.

- Improvements:

- Increased consistency in edge detection across images with different brightness levels.
- Further reduced noise, ensuring only relevant features were included in bounding boxes.

- Validation:

- Visualizations highlighted cleaner, more accurate bounding boxes with minimal false positives.

Current Version: Integrated Contour Detection

- Enhancements:

- Incorporated contour detection to extract precise boundaries of features.
- Combined all previous enhancements into a unified preprocessing pipeline.

- Improvements:

- Achieved the highest accuracy in bounding box creation.
- Enhanced performance for complex and high-noise images.

- Validation:

- Images clearly show precise, cohesive bounding boxes that match the features' actual boundaries.

Progression Over Time

1. Testing and Feedback:

- Each version was tested against a set of benchmark images with known features.
- Feedback from visualizations guided iterative refinements.

2. Key Metrics for Validation:

- Accuracy of edge detection and bounding box alignment.
- Reduction in over-segmentation and false positives.
- Consistency across diverse datasets.

3. Outcomes:

- Each iteration addressed specific limitations, resulting in a robust and reliable splicing subsystem.

By iteratively refining the Image Splicer and validating performance through visualizations, the subsystem now reliably detects and processes regions of interest with high precision. This evolution demonstrates the value of iterative development and validation in achieving a scalable and accurate image processing solution.

19.6.2 Database Validation

- Confirms data consistency between the Feature and Image Databases.
- Tests query performance and ensures efficient storage and retrieval.

19.6.3 Integration Testing

- Validates end-to-end workflows, from image upload to database updates.

19.7 Conclusion

The Database and Image Processing Subsystem, with its integration of advanced image splicing and dual-database architecture, ensures accurate feature detection and efficient data management. By automating the cleaning, splicing, classification, and storage processes, it supports scalable, real-time analysis for agricultural research. This subsystem forms the backbone of GrazeView's mission to deliver actionable insights with precision and efficiency.

19.8 Future Development

The Database and Image Processing Subsystem is currently designed with robust preprocessing and database organization capabilities. However, future developments aim to enhance its functionality by enabling communication with the machine learning (ML) model and the User Interface (UI) Data Viewer subsystem.

19.8.1 Integration with the Machine Learning Model

Currently, the system generates random feature classifications for validation purposes. Future development will focus on establishing real-time communication with the ML model:

1. CSV-Based Communication:
  - Exporting Bounding Box Data:
    - The system will generate a CSV file containing bounding box information for each subimage.
    - Each entry will include:
      - Bounding box coordinates (center, width, height).
      - Subimage references and unique IDs for traceability.
  - Sending to ML Model:
    - This CSV file will be sent to the ML model, which will analyze the bounding boxes and classify the features.

2. Receiving ML Classifications:
  - The ML model will return a CSV file with classifications for each bounding box, encoded as numeric feature types (e.g., Nale = 0, Qufu = 1, etc…).
  - The system will integrate these classifications into the Feature Database, associating them with the corresponding bounding box and source image.

3. Dynamic Database Updates:
  - Both the Feature and Image Databases will be dynamically updated:
    - Feature Database: Each bounding box entry will include the ML-assigned feature classification.
    - Image Database: Feature counts and percentage breakdowns will be appended to the database based on the ML output.

19.8.2 Interconnectivity with the UI Data Viewer

The UI Data Viewer subsystem is responsible for presenting analysis results to the user in a comprehensive and interactive manner. Future development will ensure seamless integration between the Database Subsystem and the UI Data Viewer:

1. Pulling Percentage Breakdowns:
   - The UI Data Viewer will query the Image Database to retrieve the percentage breakdown of each feature for individual images.
   - This data will be displayed alongside other metrics, enabling users to analyze feature distributions effectively.

2. Enhanced User Experience:
   - The interconnected subsystems will allow users to:
     - View high-level summaries (feature percentages) for quick insights.
   - The UI Data Viewer will support filtering, sorting, and exporting of data, empowering users with flexible analysis options.

19.8.3 Benefits of Future Development

1. Seamless Integration with ML Model:
   - Real-time communication with the ML model will eliminate the use of placeholder data, ensuring accurate and reliable feature classifications.

2. Enhanced Data Accuracy:
   - ML-driven feature detection will improve the precision of database entries, enhancing the overall system reliability.

3. Improved User Interactivity:
   - The integration with the UI Data Viewer will make feature breakdowns and bounding box data accessible and visually intuitive, enriching the user experience.

4. Scalability:
   - The CSV-based communication workflow ensures that the system can scale to handle large datasets and high-throughput ML analysis.

5. Research and Application
   - The interconnected subsystems will provide actionable insights for agricultural research, making the GrazeView application more impactful and user-friendly.