**CHAPTER 1**

**INTRODUCTION**

* 1. **PROBLEM DESCRIPTION**
* Rice as Global Staple: Rice stands as the primary food source worldwide, serving as the fundamental grain for the largest population on the planet. Particularly in tropical Asia, it holds the status of staple food, providing the primary dietary energy and protein for the majority of the population.
* Importance of Quality Paddy: The selection of high-quality paddy is crucial for ensuring both the quantity and quality of milled rice. Immature paddy, due to its chalky nature, can result in broken rice during milling processes, emphasizing the importance of proper selection.
* Pest Population Density Assessment: Assessing the density of pests in rice fields is critical for making informed pest management decisions. Sticky traps are commonly used to capture insect pests, which are then manually counted and identified in laboratories. However, this manual process is time-consuming, prone to inaccuracies, and can lead to delays in implementing effective pest management strategies.
* Integration of Image Processing: The advancement of digital technology, specifically image processing techniques, offers a solution to this complex issue. Implementing image analysis allows for the automation of insect pest detection, enabling the development of an automated system to estimate pest densities in rice fields.
* Benefits of Automation: With this automated system, crop technicians can simplify the pest counting process, enabling more efficient and accurate monitoring of rice fields. Utilizing cameras for rice infestation detection provides an easier and quicker means of monitoring.
* Role in Rice Grading: Beyond pest management, computer vision has been instrumental in rice grading. Traditionally, manual and visual methods were employed, but these are subjective and prone to errors due to human limitations. Automated systems using image processing technologies offer more objective and reliable means of assessing rice quality, thereby influencing its grade and subsequent market price.
* By integrating image processing technology, the rice production industry can benefit from more efficient pest management practices and improved rice quality assessment, ultimately contributing to increased production quantity and enhanced overall quality.

**1.2 OBJECTIVE**

This project's main goal is to combine the Convolutional Neural Network (CNN) algorithm with cutting-edge image processing techniques to create and execute a complete rice quality prediction system. With this project, we hope to solve the difficulties in manually evaluating the quality of rice and present an automated method that yields accurate and objective findings. The system's image processing part will extract important characteristics, like grain size, shape, colour, and texture, from photographs of rice. By employing methods such as picture segmentation, feature extraction, and pattern recognition, our goal is to generate a solid depiction of the visual features linked to various rice quality indicators. This project's main objective is to aid in the improvement of rice quality control procedures in the food and agriculture sectors. The automated prediction system can improve overall production and distribution efficiency of premium rice products by streamlining quality evaluation processes and lowering the subjectivity of manual inspections. It also has the potential to reduce financial losses caused by subpar quality and help those involved in the rice supply chain make better decisions.

**1.3 SCOPE**

This project's scope goes beyond the simple installation of a rice quality prediction system; it also has wider ramifications for the food and agriculture sectors. Our goal is to transform the rice quality assessment area by creating an automated solution that leverages the capability of the Convolutional Neural Network (CNN) algorithm in conjunction with image processing techniques. By integrating this technology into currently operating rice processing facilities, the need for manual inspections might be greatly reduced since rice samples could be evaluated objectively and in real-time. Another facet of the created solution's scope is its scalability. The same framework and ideas might be applied to other agricultural goods as it demonstrates its effectiveness in predicting rice quality, helping to further the larger paradigm shift towards automation and precision agriculture. Furthermore, by using deep learning models like CNN, the system may be continuously improved and adjusted to meet changing market needs and quality requirements. This is made possible by the opening of new research and development paths.

**CHAPTER 2**

**LITERATURE SURVEY**

**2.1 TITLE: SPATIAL AND SPECTRAL FEATURES UTILIZATION ON A HYPERSPECTRAL IMAGING SYSTEM FOR RICE SEED VARIETAL PURITY INSPECTION**

**AUTHOR: HAI VU, CHRISTOS TACHTATZIS**

Ensuring rice seed quality is a significant challenge for the large rice export nations such as India, Thailand, US and Vietnam. Rice seed impurities can impact on the yield by introducing weeds and off-types into the crop making it susceptible to disease. The consequences are not limited to a decrease in yield but also to the grade and price of the produce. A responsibility lies with rice seed producers to ensure high quality seed and a critical procedure is the batch screening and inspection. Conventional methods to inspect seeds, rely on extracting a sample from a batch and human visual inspection. The inspection of the sample is performed visually to assess the grain properties, such as shape, length, width and size. This task is tedious, laborious, time consuming and requires trained and experienced personnel. A conventional method to inspect the varietal purity of rice seeds is based on evaluating human visual inspection where a random sample is drawn from a batch. This is a tedious, laborious, time consuming and extremely inefficient task. This paper presents an automatic rice seed inspection method using Hyperspectral imaging and machine learning, to automatically detect unwanted seeds from other varieties which may be contained in a batch. Hyper spectral image data from Near-infrared (NIR) and Visible cameras are acquired for six common rice seed varieties. The results of applying two classifiers are presented, a Support Vector Machine (SVM) and a Random Forest (RF), where each consists of six one-versus-rest binary classifiers. The results show that combining spectral and shapebased features derived from the rice seeds, increase precision of the multi-label classification to 84% compared 74% when only visual features are used.

**2.2 TITLE: A NOVEL METHOD OF IDENTIFYING PADDY SEED VARIETIES**

**AUTHOR: KUO-YI HUANG AND MAO-CHIEN CHIEN**

Paddy is one of the main crops in Taiwan and can be planted twice each year. Purity analysis is crucial for nurseries and farmers, and purity is determined by paddy variety inspection. Purity is defined by professional inspectors according to the paddy’s appearance, shape, and color. However, there are approximately 500 cases (every case including approximately 4000 paddy seeds) of incorrect purity analyses every year in Taiwan. The jobs of inspection burden the inspectors with loading. Because healthy seedlings from seedling propagation stations (nurseries) are used to cultivate fields of paddy, seed quality is a critical factor when growing seedlings.This paper presents a novel method for identifying three varieties (Taikong, Tainan, and Taikong) of foundation paddy seeds. Taikong, Tainan, and Taikong paddy seeds are indistinguishable by inspectors during seed purity inspections. The proposed method uses image segmentation and a key point identification algorithm that can segment paddy seed images and extract seed features. A back propagation neural network was used to establish a classifier based on seven features that could classify the three paddy seed varieties. The classification accuracies of the resultant classifier for varieties Taikong, Tainan, and Taikong were 92.68%, 97.35% and 96.57%, respectively. The experimental results indicated that the three paddy seeds can be differentiated efficiently using the developed system.Image processing is widely used to inspect grain. And used a scanner to capture images of five Iranian rice kernel varieties and then extracted 41 shape features using image processing. The k-nearest neighbor’s algorithm, a support vector machine, and a back propagation neural network (BPNN) were used for classification. The classifications of the support vector machine and BPNN were favorable, with accuracies of 97% and 96%, respectively. And using used a least-squares classifier to identify five varieties of grain through their shape and color features, which were extracted using image processing. The average accuracy of classification was 99.6%. And used image processing and sparse-representation-based classification to distinguish between 30 varieties of rice grains.

**2.3 TITLE: MODELING SHAPES USING UNIFORM CUBIC B-SPLINES FOR RICE SEED IMAGE ANALYSIS**

**AUTHOR: CAROLINE NATALIE M. PERALTA**

A piecewise model for representing and matching rice seed shapes is presented to be used for visual phenotyping of rice accessions. Scanned seed images for an accession are analyzed to characterize its seeds’ shape. The resulting shape is modeled using uniform cubic B-splines, and saved to a database of representative splines for each accession. Databases using uniform cubic B-Splines with four, eight, 12, 16, and 20 curve segments were generated, and two classifiers for matching new, unknown shapes to the known accession shapes were tested. Based on tests, using uniform cubic B-Splines with four curve segments yielded promising results, and using eight, 12, 16, or 20 curve segments did not improve the classifiers’ accuracy significantly. The rice seed classification problem is addressed using uniform cubic B-Splines, a type of piecewise cubic polynomial function that can be used to fit curves. The maximum number of control points to be considered was determined by observing the trend of residual error from the original curve to the B-Spline model. Uniform cubic B-Spline Prototype Curves with four, eight, 12, 16, and 20 control points each, were then generated, and 90 test cases with ten accessions each were generated to test the performance of classification using residual error and average point distance. Both classifiers yielded poor results when tested against the entire sample of accessions, but when the accessions were narrowed to the well accessions that were observed to be reasonably identified by the classifier, the results were more promising, when both classifiers averaged an accuracy measure greater than 50%: 65.04% for the residual error classifier, and 64.656% for the average point distance classifier.

**2.4 TITLE: A METHOD FOR RAPID IDENTIFICATION OF RICE ORIGIN BY HYPERSPECTRAL IMAGING TECHNOLOGY**

**AUTHOR: JUN SUN**

The potential of hyper spectral imaging system was evaluated for the rapid identification of rice origin. 240 samples from four different regions of China were imaged by a hyperspectral imaging system. Hyper spectral images were studied from the three principal aspects (spectral, morphological and texture features). Support vector machine was used for developing the identification models. Seven models based on spectral, morphological, texture, combined spectral and morphological, combined spectral and texture, combined morphological and texture and combined spectral, morphological and texture features were developed for seeking the optimal feature combination. Nine important wavelengths were determined by principal component analysis. The results showed that the highest accuracy (91.67%) was obtained from combined spectral, morphological and texture features. This study demonstrated that hyper spectral imaging could provide a rapid identification of rice origin and the method of feature combination could be very helpful to improve the performance of identification models.Rice is an important food ration of Chinese people, which contains a great number of starch, protein, fat and some nutrient elements for people (Shao et al. 2008). Rice from different geographical origins has different characteristics and qualities, like shapes, tastes and nutrient contents, so the price and quality of rice mainly depends on its geographical origin in the food market. With the improvement of people’s living standard, consumer demand for rice is no longer limited to feeding themselves. Rice is an important food ration of Chinese people, which contains a great number of starch, protein, fat and some nutrient elements for people (Shao et al. 2008). Rice from different geographical origins has different characteristics and qualities, like shapes, tastes and nutrient contents, so the price and quality of rice mainly depends on its geographical origin in the food market. With the improvement of people’s living standard, consumer demand for rice is no longer limited to feeding themselves

**2.5 TITLE: USE OF HYPERSPECTRAL IMAGING FOR CAKE MOISTURE AND HARDNESS PREDICTION**

**AUTHOR: ADAM POLAK**

Food production is constantly confronted with increasingly rigorous requirements on the product quality, ingredients, and shelf lifetime. In this age of mass food production, which is facilitated by a network of retailers enabling global distribution, product shelf life has become one of the critical quality factors that manufacturers must maximize and accurately predict to position them strongly in a highly competitive market. Various metrics are used to assess and track the quality of baked goods during production, but these are often based on various statistical sampling schedules and the measurement procedures themselves are often time consuming, expensive, and destructive. The proposal of an on-line, stand-off test, monitoring 100% of produced goods is a solution that has the potential to solve many quality control issues. Industrial baking of sponge cakes requires various quality indicators to be measured during production such as moisture content and sponge hardness. Existing techniques for measuring these properties require randomly selected sponges to be removed from the production line before samples are manually cut out of each sponge in a destructive way for testing. These samples are subsequently processed manually using dedicated analyzers to measure moisture and texture properties in a lengthy process, which can take a skilled operator around 20 min to complete per sponge. In this study, the authors present a new, single sensor hyper spectral imaging approach, which has the potential to measure both sponge moisture content and hardness simultaneously. In the last decade, hyper spectral imaging systems have reduced in cost and size and, as a result, they are becoming widely used in a number of industries and research areas. Recently, there has been an increased use of this technology in the food industry and in food science applications and research. The application of this technology in the cake production environment, empowered by sophisticated signal and image processing techniques and prediction algorithms as presented in this study has the potential to provide on-line, real-time, stand-off cake quality monitoring.

**CHAPTER 3**

**SYSTEM ANALYSIS**

* 1. **EXISTING SYSTEM**

Several challenges has occurred while creating the dataset of rice-paddy, we tackled it by using meaningful information from an image. Literature Survey has given a knowledge of making own dataset of rice-paddy at industrial level. The bounding box is a rectangle drawn on the image which tightly fits the object in the image. A bounding box exists for every instance of every object in the image. For the box, 4 numbers (center x, center y, width, height) are predicted. This can be trained using a distance measure between predicted and ground truth bounding box.In the traditional method, this evaluation task is executed manually. But this manual approach is different from person to person and time-consuming. To overcome this problem we can use an image processing technique with deep learning. Some researcher used some these methods but their method is not capable to detect overlapped grain. The grain which is overlapped with one another is considered as one rice or negative quality of rice by exiting algorithm, which reduces the efficiency of an automated quality evaluation process. By using a hybrid approach of segmentation and edge detection we can detect the sub- region of an image which contains overlapped rice. After finding this sub-region we will detect individual rice by a convolutional neural network with its pixel intensity. By this approach, we can deliver an automated quality evaluation method which consumes less time and more efficient than the manual approach

**DISADVANTAGES**

* Need manual intervention to classify the rice quality
* Time complexity can be high
* Irrelevant features are extraction
* So dimensionality can be high
* Misclassification can be occurred
  1. **PROPOSED SYSTEM**

Rice has been one of the most widely consumed foods for a large part of human population. Numerous different rice varieties are imported and exported worldwide, making it the backbone of many countries’ economy. Rice seeds of different varieties can be accidentally or intentionally mixed during any of the steps in a rice production pipeline, introducing impurities. These impurities could damage the trust between rice importers and exporters, calling for the need to develop a reliable rice variety inspection system. Different rice varieties could be mixed during rice production and trading. Rice impurities could damage the trust between rice importers and exporters, calling for the need to develop a rice variety inspection system. In this work, we develop a non-destructive rice variety classification system that benefits from the synergy between imaging and deep convolutional neural network (CNN). The proposed method uses a imaging system to simultaneously acquire complementary spatial and spectral information of rice seeds. The rice varieties are then determined from the acquired spatio-spectral data using a deep CNN. As opposed to several existing rice variety classification methods that require hand-engineered features, the proposed method automatically extracts spatio-spectral features from the raw sensor data.In this work, we developed a data classification method based on a deep CNN with hundreds of processing layers. We modified and extended a residual network (ResNet) with bottleneck building blocks to enable spatio-spectral data classification. The proposed method (abbreviated as ResNet-B) takes advantages of residual connections together with bottleneck building blocks, batch normalization, and early stopping to aid the training process of such a deep network.

**ADVANTAGES**

* Automated approach
* Relevant features are extracted
* Time and computational complexity can be reduced
* Classification accuracy is high

**CHAPTER 4**

**SYSTEM SPECIFICATION**

* 1. **HARDWARE REQUIREMENTS**
* Processor : Intel core processor 2.6.0 GHZ
* RAM : 4 GB
* Hard disk : 160 GB
* Compact Disk : 650 Mb
* Keyboard : Standard keyboard
* Monitor : 15 inch color monitor
  1. **SOFTWARE REQUIREMENTS**
* Operating system : Windows OS
* Front End : .NET (C#)
* Back End : SQL SERVER
* IDE : VISUAL STUDIO

**CHAPTER 5**

**SOFTWARE DESCRIPTION**

**FRONT END: .NET FRAMEWORK**

The .NET Framework (pronounced dot net) is a [software framework](http://en.wikipedia.org/wiki/Software_framework) developed by [Microsoft](http://en.wikipedia.org/wiki/Microsoft) that runs primarily on [Microsoft Windows](http://en.wikipedia.org/wiki/Microsoft_Windows). It includes a large [library](http://en.wikipedia.org/wiki/Base_Class_Library) and provides [language interoperability](http://en.wikipedia.org/wiki/Language_interoperability) (each language can use code written in other languages) across several [programming languages](http://en.wikipedia.org/wiki/Programming_language). Programs written for the .NET Framework execute in a [software](http://en.wikipedia.org/wiki/Software) environment (as contrasted to [hardware](http://en.wikipedia.org/wiki/Computer_hardware) environment), known as the [Common Language Runtime](http://en.wikipedia.org/wiki/Common_Language_Runtime) (CLR), an [application virtual machine](http://en.wikipedia.org/wiki/Process_virtual_machine) that provides services such as security, [memory management](http://en.wikipedia.org/wiki/Memory_management), and [exception handling](http://en.wikipedia.org/wiki/Exception_handling). The class library and the CLR together constitute the .NET Framework.

The .NET Framework's [Base Class Library](http://en.wikipedia.org/wiki/Base_Class_Library) provides [user interface](http://en.wikipedia.org/wiki/User_interface), [data access](http://en.wikipedia.org/wiki/Data_access), [database connectivity](http://en.wikipedia.org/wiki/Database_connection), [cryptography](http://en.wikipedia.org/wiki/Cryptography), [web application](http://en.wikipedia.org/wiki/Web_application) development, numeric [algorithms](http://en.wikipedia.org/wiki/Algorithm), and [network communications](http://en.wikipedia.org/wiki/Computer_networking). Programmers produce software by combining their own [source code](http://en.wikipedia.org/wiki/Source_code) with the .NET Framework and other libraries. The .NET Framework is intended to be used by most new applications created for the Windows platform. Microsoft also produces an [integrated development environment](http://en.wikipedia.org/wiki/Integrated_development_environment) largely for .NET software called [Visual Studio](http://en.wikipedia.org/wiki/Microsoft_Visual_Studio)

**Design Features**

**Interoperability**

Because computer systems commonly require interaction between newer and older applications, the .NET Framework provides means to access functionality implemented in newer and older programs that execute outside the .NET environment. Access to [COM](http://en.wikipedia.org/wiki/Component_Object_Model) components is provided in the System. Runtime. Interpol Services and System. Enterprise Services namespaces of the framework; access to other functionality is achieved using the [P/Invoke](http://en.wikipedia.org/wiki/Platform_Invocation_Services) feature.

**Common Language Runtime engine**

The [Common Language Runtime](http://en.wikipedia.org/wiki/Common_Language_Runtime) (CLR) serves as the execution engine of the .NET Framework. All .NET programs execute under the supervision of the CLR, guaranteeing certain properties and behaviors in the areas of memory management, security, and exception handling.

**Language independence**

The .NET Framework introduces a [Common Type System](http://en.wikipedia.org/wiki/Common_Type_System), or CTS. The CTS [specification](http://en.wikipedia.org/wiki/Specification) defines all possible [data types](http://en.wikipedia.org/wiki/Datatypes) and [programming](http://en.wikipedia.org/wiki/Programming) constructs supported by the CLR and how they may or may not interact with each other conforming to the [Common Language Infrastructure](http://en.wikipedia.org/wiki/Common_Language_Infrastructure) (CLI) specification. Because of this feature, the .NET Framework supports the exchange of types and object instances between libraries and applications written using any conforming [.NET language](http://en.wikipedia.org/wiki/List_of_CLI_languages).

**Base Class Library**

The [Base Class Library](http://en.wikipedia.org/wiki/Base_Class_Library) (BCL), part of the Framework Class Library (FCL), is a library of functionality available to all languages using the .NET Framework. The BCL provides [classes](http://en.wikipedia.org/wiki/Class_%28computer_science%29) that encapsulate a number of common functions, including [file](http://en.wikipedia.org/wiki/Computer_file) reading and writing, [graphic rendering](http://en.wikipedia.org/wiki/Rendering_%28computer_graphics%29), [database](http://en.wikipedia.org/wiki/Database) interaction, [XML](http://en.wikipedia.org/wiki/XML) document manipulation, and so on. It consists of classes, interfaces of reusable types that integrate with CLR (Common Language Runtime).

**Simplified deployment**

The .NET Framework includes design features and tools which help manage the [installation](http://en.wikipedia.org/wiki/Installation_%28computer_programs%29) of computer software to ensure it does not interfere with previously installed software, and it conforms to security requirements.

**Security**

The design addresses some of the vulnerabilities, such as [buffer overflows](http://en.wikipedia.org/wiki/Buffer_overflow), which have been exploited by malicious software. Additionally, .NET provides a common security model for all applications.

**Portability**

While Microsoft has never implemented the full framework on any system except Microsoft Windows, it has engineered the framework to be platform-agnostic,[[3]](http://en.wikipedia.org/wiki/.NET_Framework#cite_note-3) and cross-platform implementations are available for other operating systems (see [Silverlight](http://en.wikipedia.org/wiki/Silverlight) and the [Alternative implementations](http://en.wikipedia.org/wiki/.NET_Framework#Alternative_implementations) section below). Microsoft submitted the specifications for the [Common Language Infrastructure](http://en.wikipedia.org/wiki/Common_Language_Infrastructure) (which includes the core class libraries, [Common Type System](http://en.wikipedia.org/wiki/Common_Type_System), and the [Common Intermediate Language](http://en.wikipedia.org/wiki/Common_Intermediate_Language)), the [C#](http://en.wikipedia.org/wiki/C_Sharp_%28programming_language%29) language, and the C++/CLI language[[8]](http://en.wikipedia.org/wiki/.NET_Framework#cite_note-8) to both [ECMA](http://en.wikipedia.org/wiki/Ecma_International) and the [ISO](http://en.wikipedia.org/wiki/International_Organization_for_Standardization), making them available as official standards. This makes it possible for third parties to create compatible implementations of the framework and its languages on other platforms.

**Common Language Infrastructure (CLI)**

The purpose of the Common Language Infrastructure (CLI) is to provide a language-neutral platform for application development and execution, including functions for [Exception handling](http://en.wikipedia.org/wiki/Exception_handling), [Garbage Collection](http://en.wikipedia.org/wiki/Garbage_collection_%28computer_science%29), security, and interoperability. By implementing the core aspects of the .NET Framework within the scope of the CL, this functionality will not be tied to a single language but will be available across the many languages supported by the framework. Microsoft's implementation of the CLI is called the [Common Language Runtime](http://en.wikipedia.org/wiki/Common_Language_Runtime), or CLR.

The [CIL](http://en.wikipedia.org/wiki/Common_Intermediate_Language) code is housed in [CLI assemblies](http://en.wikipedia.org/wiki/Assembly_%28CLI%29). As mandated by the specification, assemblies are stored in the [Portable Executable](http://en.wikipedia.org/wiki/Portable_Executable) (PE) format, common on the Windows platform for all [DLL](http://en.wikipedia.org/wiki/Dynamic-link_library) and [EXE](http://en.wikipedia.org/wiki/EXE) files. The assembly consists of one or more files, one of which must contain the manifest, which has the [metadata](http://en.wikipedia.org/wiki/Metadata) for the assembly. The complete name of an assembly (not to be confused with the filename on disk) contains its simple text name, version number, culture, and [public key](http://en.wikipedia.org/wiki/Public_key) token. Assemblies are considered equivalent if they share the same complete name, excluding the revision of the version number. A private key can also be used by the creator of the assembly for strong naming. The public key token identifies which public key an assembly is signed with. Only the creator of the keypair (typically the .NET developer signing the assembly) can sign assemblies that have the same strong name as a previous version assembly, since he is in possession of the private key. [Strong naming](http://en.wikipedia.org/wiki/Strong_name) is required to add assemblies to the [Global Assembly Cache](http://en.wikipedia.org/wiki/Global_Assembly_Cache)

**Security**

.NET has its own security mechanism with 2 general features: [Code Access Security](http://en.wikipedia.org/wiki/Code_Access_Security) (CAS), and validation and verification. Code Access Security is based on evidence that is associated with a specific assembly. Typically the evidence is the source of the assembly (whether it is installed on the local machine or has been downloaded from the intranet or Internet). Code Access Security uses evidence to determine the permissions granted to the code. Other code can demand that calling code is granted a specified permission. The demand causes the [CLR](http://en.wikipedia.org/wiki/Common_Language_Runtime) to perform a call stack walk: every assembly of each method in the call stack is checked for the required permission; if any assembly is not granted the permission a security exception is thrown.

**Class library**

The .NET Framework includes a set of [standard](http://en.wikipedia.org/wiki/Standard_library) class libraries. The class library is organized in a hierarchy of [namespaces](http://en.wikipedia.org/wiki/Namespace_%28computer_science%29). Most of the built-in APIs are part of either System.\* or Microsoft.\* namespaces. These class libraries implement a large number of common functions, such as file reading and writing, graphic rendering, database interaction, and XML document manipulation, among others. The .NET class libraries are available to all [CLI compliant languages](http://en.wikipedia.org/wiki/List_of_CLI_languages). The .NET Framework class library is divided into two parts: the Base Class Library and the Framework Class Library

The [Base Class Library](http://en.wikipedia.org/wiki/Base_Class_Library) (BCL) includes a small subset of the entire class library and is the core set of classes that serve as the basic [API](http://en.wikipedia.org/wiki/API) of the [Common Language Runtime](http://en.wikipedia.org/wiki/Common_Language_Runtime).[[9]](http://en.wikipedia.org/wiki/.NET_Framework#cite_note-bcllibs-9) The classes in mscorlib.dll and some of the classes in System.dll and System.core.dll are considered to be a part of the BCL. The BCL classes are available in both .NET Framework as well as its alternative implementations including [.NET Compact Framework](http://en.wikipedia.org/wiki/.NET_Compact_Framework), [Microsoft Silverlight](http://en.wikipedia.org/wiki/Microsoft_Silverlight) and [Mono](http://en.wikipedia.org/wiki/Mono_%28software%29).

The [Framework Class Library](http://en.wikipedia.org/wiki/Framework_Class_Library) (FCL) is a superset of the BCL classes and refers to the entire class library that ships with .NET Framework. It includes an expanded set of libraries, including [Windows Forms](http://en.wikipedia.org/wiki/Windows_Forms), [ADO.NET](http://en.wikipedia.org/wiki/ADO.NET), [ASP.NET](http://en.wikipedia.org/wiki/ASP.NET), [Language Integrated Query](http://en.wikipedia.org/wiki/Language_Integrated_Query), [Windows Presentation Foundation](http://en.wikipedia.org/wiki/Windows_Presentation_Foundation), [Windows Communication Foundation](http://en.wikipedia.org/wiki/Windows_Communication_Foundation) among others. The FCL is much larger in scope than standard libraries for languages like [C++](http://en.wikipedia.org/wiki/C%2B%2B), and comparable in scope to the [standard libraries of Java](http://en.wikipedia.org/wiki/Java_Class_Library).

**Memory management**

The .NET Framework CLR frees the developer from the burden of managing memory (allocating and freeing up when done); it handles memory management itself by detecting when memory can be safely freed. Memory is allocated instantiations of .NET types (objects) from the managed heap, a pool of memory managed by the CLR. As long as there exists a reference to an object, which might be either a direct reference to an object or via a [graph](http://en.wikipedia.org/wiki/Graph_%28data_structure%29) of objects, the object is considered to be in use. When there is no reference to an object, and it cannot be reached or used, it becomes garbage, eligible for collection. NET Framework includes a [garbage collector](http://en.wikipedia.org/wiki/Garbage_collection_%28computer_science%29) which runs periodically, on a separate [thread](http://en.wikipedia.org/wiki/Thread_%28computing%29) from the application's thread, that enumerates all the unusable objects and reclaims the memory allocated to them.

The .NET [Garbage Collector](http://en.wikipedia.org/wiki/Garbage_collection_%28computer_science%29) (GC) is a non-deterministic, compacting, [mark-and-sweep](http://en.wikipedia.org/wiki/Garbage_collection_%28computer_science%29#Copying_vs._mark-and-sweep_vs._mark-and-don.27t-sweep) garbage collector. The GC runs only when a certain amount of memory has been used or there is enough pressure for memory on the system. Since it is not guaranteed when the conditions to reclaim memory are reached, the GC runs are non-deterministic. Each .NET application has a set of roots, which are pointers to objects on the managed heap (managed objects). These include references to static objects and objects defined as local variables or method parameters currently in scope, as well as objects referred to by CPU registers.[[10]](http://en.wikipedia.org/wiki/.NET_Framework#cite_note-part1-10) When the GC runs, it pauses the application, and for each object referred to in the root, it [recursively](http://en.wikipedia.org/wiki/Recursion) enumerates all the objects reachable from the root objects and marks them as reachable. It uses CLI metadata and [reflection](http://en.wikipedia.org/wiki/Reflection_%28computer_science%29) to discover the objects encapsulated by an object, and then recursively walk them. It then enumerates all the objects on the heap (which were initially allocated contiguously) using reflection. All objects not marked as reachable are garbage. This is the mark phase. Since the memory held by garbage is not of any consequence, it is considered free space. However, this leaves chunks of free space between objects which were initially contiguous. The objects are then compacted together to make used memory contiguous again. Any reference to an object invalidated by moving the object is updated by the GC to reflect the new location. The application is resumed after the garbage collection is over.

**CHAPTER 6**

**PROJECT DESCRIPTION**

**6.1 MODULES**

* Image Upload
* Preprocessing
* Edge Detection
* Features Extraction
* Rice Classification

**6.2 MODULES DESCRIPTION**

**6.2.1 IMAGE UPLOAD**

In agricultural industry quality analysis of product is very important. Quality of grain seeds is analyzed visually by experienced technician. But the outcome of such measurement is relative, varying in results and time consuming. The quality also gets affected by the mood of technician; so to overcome the shortcomings occurred due to traditional methods new and advanced technique i.e. image processing technique is proposed. Image Acquisition is the first step in image processing. Acquisition is done by using data sets under uniform lighting setup. The captured image stored in the desktop using USB cable. After storing the image on desktop image processing algorithms are applied on it.

**6.2.2 PREPROCESSING**

The aim of pre-processing is an improvement of the image data that suppresses unwilling distortions or enhances some image features important for further processing, although geometric transformations of images (e.g. rotation, scaling, and translation) are classified among pre-processing methods here since similar techniques are used. The user has to select the required lung frame image for further processing. Then each image is resized to 256\*256. Then implement median filter to remove noises from lung images.

**6.2.3 EDGE DETECTION**

Edge detection is based on recognition of edges by diverse edge operators. Discontinuities in color, Grey level, texture, etc. are detected by edge operators. The Sobel operator, sometimes called the Sobel–Feldman operator or Sobel filter, is used in [image processing](https://en.wikipedia.org/wiki/Image_processing) and [computer vision](https://en.wikipedia.org/wiki/Computer_vision), particularly within [edge detection](https://en.wikipedia.org/wiki/Edge_detection) algorithms where it creates an image emphasising edges. Since the intensity function of a digital image is only known at discrete points, derivatives of this function cannot be defined unless we assume that there is an underlying continuous intensity function which has been sampled at the image points.

**6.2.4 FEATURES EXTRACTION**

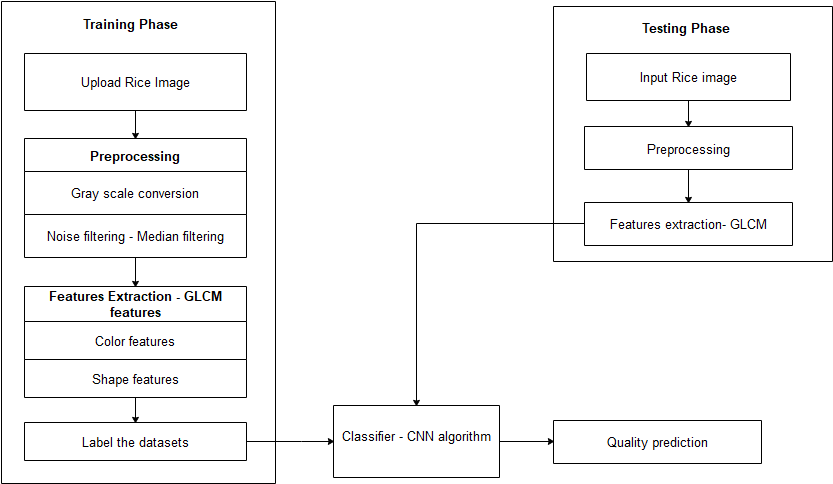
Feature extraction involves simplifying the amount of resources required to describe a large set of data accurately. When performing analysis of complex data one of the major problems stems from the number of variables involved. Analysis with a large number of variables generally requires a large amount of memory and computation power or a classification algorithm which over fits the training sample and generalizes poorly to new samples. Feature extraction is a general term for methods of constructing combinations of the variables to get around these problems while still describing the data with sufficient accuracy.

**6.2.5 RICE CLASSIFICATION**

A Convolutional neural network (CNN) algorithm is a new pattern classifier was trained for classification of the samples into the grades. CNN models are closely related. The technique has successfully been applied to standard classification tasks, such as text classification and medical diagnosis. CNNs avoid the “curse of dimensionality” by placing an upper bound on the margin between the different classes, making it a practical tool for large, dynamic datasets. The feature space may even be reduced further by selecting the most distinguishing features through minimization of the feature set size.

* 1. **BLOCK DIAGRAM**

A system architecture or systems architecture is the conceptual model that defines the structure, behavior, and more views of a system. An architecture description is a formal description and representation of a system, organized in a way that supports reasoning about the structures and behaviors of the system. System architecture can comprise system components, the externally visible properties of those components, the relationships (e.g. the behavior) between them. It can provide a plan from which products can be procured, and systems developed, that will work together to implement the overall system. There have been efforts to formalize languages to describe system architecture; collectively these are called architecture description languages (ADLs).

****

**CHAPTER 7**

**IMPLEMENTATION AND RESULT ANALYSIS**

Implementation is the stage in the project where the theoretical design is turned into a working system. The most critical stage is achieving a successful system and in giving confidence on the new system for the users, what it will work efficient and effectively. It involves careful planning, investing of the current system, and its constraints on implementation, design of methods to achieve the change over methods. The implementation process begins with preparing a plan for the implementation of the system. According to this plan, the activities are to be carried out in these plans; discussion has been made regarding the equipment, resources and how to test activities. The coding step translates a detail design representation into a programming language

Realization. Programming languages are vehicles for communication between human and computers programming language characteristics and coding style can profoundly affect software quality and maintainability. The coding is done with the following characteristics in mind.

* Ease of design to code translation.
* Code efficiency.
* Memory efficiency.
* Maintainability.

Implementation is the stage of the project when the theoretical design is turned out into a working system. Thus it can be considered to be the most critical stage in achieving a successful new system and in giving the user, confidence that the new system will work and be effective.

**CHAPTER 8**

**CONCLUSION AND FUTURE ENHANCEMENT**

Physical rice grain composition is one of the components used for rice grading which involve the identification of head rice and broken rice. Rice grading is important to ensure only edible rice reaches the consumer standard. It also protects consumers from price manipulation. In this project, a new approach of image processing technique has been developed to detect and identify head and broken rice based on its physical properties, i.e. area, perimeter, minor axis length and major axis length. From this research, it can be concluded that the physical rice grain composition can be determined using an image processing technique. Area has been identified as the most appropriate properties when compared to perimeter, major axis length and minor axis length. The method gives higher percentage of success when tested using 0%, 1%, 10%, 15% and 20% of broken rice with the average percentage of success of 98%. This promising result provides an alternative way to grade rice. The method has an advantage on its simplicity. This straightforward method is easy to be used and fast. It does not involve any step-by-step procedure to optimize the criterion and only takes 4 s to complete the operation. Although the percentage of success is promising, however, more sophisticated method might be considered to solve problem on undetected broken rice. It can be done by using more than one parameter such as combination of area and major axis length.

In future we can extend the approach to implement the neural network algorithm to predict the quality of rice with improved accuracy rate.

**CHAPTER 9**

**APPENDICES**

**9.1 SOURCE CODE**

**HOME**

using System;

usingSystem.Collections.Generic;

usingSystem.ComponentModel;

usingSystem.Data;

usingSystem.Drawing;

usingSystem.Linq;

usingSystem.Text;

usingSystem.Windows.Forms;

namespaceHerbal\_Leaf\_Classification

{

publicpartialclassForm1 : Form

{

public Form1()

{

InitializeComponent();

}

privatevoid button1\_Click(object sender, EventArgs e)

{

Homehh = newHome();

hh.Show();

}

privatevoid button2\_Click(object sender, EventArgs e)

{

}

privatevoid label1\_Click(object sender, EventArgs e)

{

}

privatevoid button2\_Click\_1(object sender, EventArgs e)

{

UploadImageuu = newUploadImage();

uu.Show();

}

}

}

**UPLOAD IMAGE**

using System;

usingSystem.Collections.Generic;

usingSystem.ComponentModel;

usingSystem.Data;

usingSystem.Drawing;

usingSystem.Linq;

usingSystem.Text;

usingSystem.Windows.Forms;

namespaceHerbal\_Leaf\_Classification

{

publicpartialclassUploadImage : Form

{

publicUploadImage()

{

InitializeComponent();

}

privatevoidUploadImage\_Load(object sender, EventArgs e)

{

}

publicstring name, ff;

privatevoid button1\_Click(object sender, EventArgs e)

{

OpenFileDialog op = newOpenFileDialog();

op.ShowDialog();

if (op.FileName == "")

{

MessageBox.Show("Please Choose Image");

}

else

{

pictureBox1.Image = newBitmap(op.FileName);

name = op.FileName;

ff = System.IO.Path.GetFileName(op.FileName);

}

}

privatevoid button3\_Click(object sender, EventArgs e)

{

if (pictureBox1.Image == null)

{

MessageBox.Show("Please Choose Image");

}

else

{

UPreprocessing p = newUPreprocessing();

p.bmp = newBitmap(pictureBox1.Image);

p.original = name;

p.ff = ff;

p.Show();

}

}

}

}

**PREPROCESSING**

using System;

usingSystem.Collections.Generic;

usingSystem.ComponentModel;

usingSystem.Data;

usingSystem.Drawing;

usingSystem.Linq;

usingSystem.Text;

usingSystem.Windows.Forms;

usingSystem.Drawing.Imaging;

usingAForge;

usingAForge.Imaging.Filters;

namespaceHerbal\_Leaf\_Classification

{

publicpartialclassUPreprocessing : Form

{

privateSystem.Drawing.Bitmapm\_Bitmap;

privateSystem.Drawing.Bitmap m\_Bitmap1;

privateSystem.Drawing.Bitmap m\_Bitmap2;

privateSystem.Drawing.Bitmap m\_Bitmap3;

privateSystem.Drawing.Bitmapm\_Undo;

privateSystem.Drawing.Bitmap m\_Undo1;

privateSystem.Drawing.Bitmap m\_Undo2;

publicBitmap bmp;

publicstring original;

publicdecimal a, b, c;

publicstringff;

publicUPreprocessing()

{

InitializeComponent();

}

privatevoidUPreprocessing\_Load(object sender, EventArgs e)

{

pictureBox1.Image = Bitmap.FromFile(original);

pictureBox1.SizeMode = PictureBoxSizeMode.StretchImage;

m\_Bitmap = (Bitmap)Bitmap.FromFile(original, false);

m\_Bitmap1 = (Bitmap)Bitmap.FromFile(original, false);

m\_Bitmap2 = (Bitmap)Bitmap.FromFile(original, false);

m\_Bitmap3 = (Bitmap)Bitmap.FromFile(original, false);

}

privatevoid button1\_Click(object sender, EventArgs e)

{

m\_Undo = (Bitmap)m\_Bitmap.Clone();

if (BitmapFilter.GrayScale(m\_Bitmap))

pictureBox1.Image = m\_Bitmap;

}

privatevoid button2\_Click(object sender, EventArgs e)

{

m\_Undo1 = (Bitmap)m\_Bitmap1.Clone();

// if (BitmapFilter.GrayScale(m\_Bitmap))

// if (BitmapFilter.Invert(m\_Bitmap))

// pictureBox1.Image = m\_Bitmap;

if (BitmapFilter.GrayScale(m\_Undo1)) { }

Median filter = newMedian();

// apply the filter

filter.ApplyInPlace(m\_Undo1);

pictureBox1.Image = newBitmap(m\_Undo1);

}

privatevoid button4\_Click(object sender, EventArgs e)

{

UFeature\_Extract f = newUFeature\_Extract();

f.orginal = original;

f.ff = ff;

f.Show();

}

}

}

**FEATURES EXTRACTION**

using System;

usingSystem.Collections.Generic;

usingSystem.ComponentModel;

usingSystem.Data;

usingSystem.Drawing;

usingSystem.Linq;

usingSystem.Text;

usingSystem.Windows.Forms;

usingAForge;

usingAForge.Imaging;

usingAForge.Math;

usingAForge.Imaging.Filters;

usingimage = System.Drawing.Image;

usingpoints = System.Drawing.Point;

namespaceHerbal\_Leaf\_Classification

{

publicpartialclassFeature\_Extract : Form

{

staticint[] BallX, BallY;

List<points> points1 = newList<points>();

publicstringorginal, ff;

publicdecimal a, b, c;

publicFeature\_Extract()

{

InitializeComponent();

}

privatevoidFeature\_Extract\_Load(object sender, EventArgs e)

{

pictureBox1.Image = newBitmap(orginal);

}

privatevoid button1\_Click(object sender, EventArgs e)

{

Bitmap image = newBitmap(pictureBox1.Image);

BitmapgrayImage;

Bitmap bmp = newBitmap(pictureBox1.Image);

Grayscale filter = newGrayscale(0.2125, 0.7154, 0.0721);

// apply the filter

grayImage = filter.Apply(bmp);

ResizeNearestNeighborrb = newResizeNearestNeighbor(pictureBox1.Width, pictureBox1.Height);

// apply the filter

image = rb.Apply(image);

//MessageBox.Show(image.Height.ToString());

MessageBox.Show(image.Width.ToString());

Graphicsgraphics = Graphics.FromImage(image);

SolidBrush brush = newSolidBrush(Color.Red);

Penpen = newPen(brush);

// Create corner detector and have it process the image

MoravecCornersDetector mcd = newMoravecCornersDetector();

List<IntPoint> corners = mcd.ProcessImage(image);

BallX = newint[Convert.ToInt32(corners.Count)];

BallY = newint[Convert.ToInt32(corners.Count)];

// Visualization: Draw 3x3 boxes around the corners

intvall = 0;

foreach (IntPoint corner in corners)

{

graphics.DrawRectangle(pen, corner.X - 1, corner.Y - 1, 1, 1);

listBox1.Items.Add("(" + corner.X.ToString() + "," + corner.Y.ToString() + ")");

BallX[vall] = corner.X;

BallY[vall] = corner.Y;

vall += 1;

//points1.Add(new points(corner.X, corner.Y));

}

// Display

pictureBox1.Image = image;

}

privatevoid button2\_Click(object sender, EventArgs e)

{

Trainingtt = newTraining();

tt.orginal = orginal;

tt.ff = ff;

tt.Show();

}

privatevoid button3\_Click(object sender, EventArgs e)

{

Colorfeature cc = newColorfeature();

cc.imagename = orginal;

cc.Show();

}

}

}

**COLORFEATURE**

using System;

usingSystem.Collections.Generic;

usingSystem.ComponentModel;

usingSystem.Data;

usingSystem.Drawing;

usingSystem.Linq;

usingSystem.Text;

usingSystem.Windows.Forms;

usingSystem.Windows.Forms.DataVisualization.Charting;

namespaceHerbal\_Leaf\_Classification

{

publicpartialclassColorfeature : Form

{

publicColorfeature()

{

InitializeComponent();

}

publicBitmap bmp;

publicBitmap bmp1;

publicstringklm;

publicBitmapbmmp;

publicstringimagename;

publicdecimal ac1, ac2, ac3, er1, er2, er3;

publicDataTableDataResults = newDataTable("Items");

publicDataTable DataResults1 = newDataTable("Items1");

publicDataTable DataResults2 = newDataTable("Items2");

privatevoidColorfeature\_Load(object sender, EventArgs e)

{

bmp = newBitmap(imagename);

bmp1 = newBitmap(imagename);

klm = "Histogram";

pictureBox1.Image = bmp;

// AForge.Math.HistogramactiveHistogram = null;

// AForge.Imaging.ImageStatistics stat =

//new AForge.Imaging.ImageStatistics(bmp);

// if (stat != null)

// {

// //Do if the pic is gray

// if (stat.IsGrayscale)

// {

// activeHistogram = stat.Red;

// }

// //Do if the pic is colourful

// if (!stat.IsGrayscale)

// {

// activeHistogram = stat.Red;

// }

// }

kk();

kk1();

kk2();

}

publicvoidkk()

{

AForge.Math.HistogramactiveHistogram = null;

AForge.Imaging.ImageStatistics stat =

newAForge.Imaging.ImageStatistics(bmp1);

if (stat != null)

{

//Do if the pic is gray

if (stat.IsGrayscale)

{

activeHistogram = stat.Red;

}

//Do if the pic is colourful

if (!stat.IsGrayscale)

{

activeHistogram = stat.Red;

}

}

DataColumndcItemValue = newDataColumn("Name");

DataColumn dcItemN1 = newDataColumn("Values");

//dcItemN1.DataType = System.Type.GetType("System.Int32");

DataResults1.Columns.Add(dcItemValue);

DataResults1.Columns.Add(dcItemN1);

//System .Int32[]

//DataResults.Rows.Add("K-Means", ac1);

//DataResults.Rows.Add("Fuzzy K-Means", ac2);

// DataResults.Rows.Add("Adaptive Fuzzy K-Means", activeHistogram.Values);

int i = 0;

foreach (intvalinactiveHistogram.Values)

{

// Console.WriteLine(val);

i++;

DataResults1.Rows.Add(i, val);

}

//chart1.DataSource = DataResults.Tables["salary"];

chart2.Series["Red"].XValueMember = "Name";

chart2.Series["Red"].YValueMembers = "Values";

this.chart2.Titles.Add("Histogram Of Red Plane");

chart2.Series["Red"].ChartType = SeriesChartType.Column;

//chart1.Series["accuracy"].IsValueShownAsLabel = true;

chart2.DataSource = DataResults1;

}

publicvoid kk1()

{

AForge.Math.HistogramactiveHistogram = null;

AForge.Imaging.ImageStatistics stat =

newAForge.Imaging.ImageStatistics(bmp1);

if (stat != null)

{

//Do if the pic is gray

if (stat.IsGrayscale)

{

activeHistogram = stat.Green;

}

//Do if the pic is colourful

if (!stat.IsGrayscale)

{

activeHistogram = stat.Green;

}

}

//histogram1.Values = activeHistogram.Values;

// histogram1.Color = System.Drawing.Color.Green;

//oo.Dispose();

//histogram1.Refresh();

DataColumndcItemValue = newDataColumn("Name");

DataColumn dcItemN1 = newDataColumn("Values");

//dcItemN1.DataType = System.Type.GetType("System.Int32");

DataResults2.Columns.Add(dcItemValue);

DataResults2.Columns.Add(dcItemN1);

//System .Int32[]

//DataResults.Rows.Add("K-Means", ac1);

//DataResults.Rows.Add("Fuzzy K-Means", ac2);

// DataResults.Rows.Add("Adaptive Fuzzy K-Means", activeHistogram.Values);

int i = 0;

foreach (intvalinactiveHistogram.Values)

{

// Console.WriteLine(val);

i++;

DataResults2.Rows.Add(i, val);

}

//chart1.DataSource = DataResults.Tables["salary"];

chart3.Series["Green"].XValueMember = "Name";

chart3.Series["Green"].YValueMembers = "Values";

this.chart3.Titles.Add("Histogram Of Green Plane");

chart3.Series["Green"].ChartType = SeriesChartType.Column;

//chart1.Series["accuracy"].IsValueShownAsLabel = true;

chart3.DataSource = DataResults2;

}

publicvoid kk2()

{

AForge.Math.HistogramactiveHistogram = null;

AForge.Imaging.ImageStatistics stat =

newAForge.Imaging.ImageStatistics(bmp1);

if (stat != null)

{

//Do if the pic is gray

if (stat.IsGrayscale)

{

activeHistogram = stat.Blue;

}

//Do if the pic is colourful

if (!stat.IsGrayscale)

{

activeHistogram = stat.Blue;

}

}

// histogram3.Values = activeHistogram.Values;

// histogram3.Color = System.Drawing.Color.Blue;

// histogram3.Values

//oo.Dispose();

// histogram3.Refresh();

DataColumn dcItemValue1 = newDataColumn("Name1");

DataColumn dcItemN11 = newDataColumn("Values1");

//dcItemN1.DataType = System.Type.GetType("System.Int32");

DataResults.Columns.Add(dcItemValue1);

DataResults.Columns.Add(dcItemN11);

//System .Int32[]

//DataResults.Rows.Add("K-Means", ac1);

//DataResults.Rows.Add("Fuzzy K-Means", ac2);

// DataResults.Rows.Add("Adaptive Fuzzy K-Means", activeHistogram.Values);

int i=0;

foreach (intvalinactiveHistogram.Values)

{

// Console.WriteLine(val);

i++;

DataResults.Rows.Add(i, val);

}

//chart1.DataSource = DataResults.Tables["salary"];

chart1.Series["Blue"].XValueMember = "Name1";

chart1.Series["Blue"].YValueMembers = "Values1";

this.chart1.Titles.Add("Histogram Of Blue Plane");

chart1.Series["Blue"].ChartType = SeriesChartType.Column;

//chart1.Series["accuracy"].IsValueShownAsLabel = true;

chart1.DataSource = DataResults;

}

privatevoid button1\_Click(object sender, EventArgs e)

{

}

privatevoid panel2\_Paint(object sender, PaintEventArgs e)

{

}

}

}

**NEURAL**

using System;

usingSystem.Collections.Generic;

usingSystem.ComponentModel;

usingSystem.Data;

usingSystem.Drawing;

usingSystem.Linq;

usingSystem.Text;

usingSystem.Windows.Forms;

using System.IO;

usingSystem.Data.SqlClient;

usingSystem.Drawing.Imaging;

usingSystem.Collections;

namespaceHerbal\_Leaf\_Classification

{

publicpartialclassNeural : Form

{

intlocX = 20;

intlocY = 10;

intsizeWidth = 160;

intsizeHeight = 160;

string path, resultpath;

byte[] photo\_aray;

BitmapSourceImage;

string[] ImagesToSearch;

publicBitmap bmp;

publicBitmap bmp1;

publicDataTableDataResults = newDataTable("Items");

SqlConnection con = newSqlConnection(@"Data Source=.\SQLEXPRESS;AttachDbFilename=C:\Users\Rajiya\Downloads\RiceQualityPrediction\RiceQualityPrediction\RiceQualityPrediction\ricetb.mdf;Integrated Security=True;User Instance=True");

SqlCommandcmd;

public Neural()

{

InitializeComponent();

}

privatevoid button1\_Click(object sender, EventArgs e)

{

FolderBrowserDialog f = newFolderBrowserDialog();

f.ShowDialog();

if (f.ShowDialog() == DialogResult.OK)

{

intAlreadyexist = 0;

for (int i = 1; i <= listBox2.Items.Count; i++)

{

listBox2.SetSelected(i - 1, true);

if (f.SelectedPath == listBox2.Text.ToString())

{

Alreadyexist = 1;

MessageBox.Show("Folder " + f.SelectedPath.ToString() + " Already Exist In The List", "Demo", MessageBoxButtons.OK, MessageBoxIcon.Asterisk);

}

}

if (Alreadyexist == 0)

{

DirectoryInfodir = newDirectoryInfo((string)f.SelectedPath.ToString());

FileInfo[] bmpfiles = dir.GetFiles("\*.jpg");

if (bmpfiles.Length>= 1)

{

listBox2.Items.Add(f.SelectedPath);

checkforimagesinfolders();

}

else

MessageBox.Show("Folder " + f.SelectedPath.ToString() + " Does Not Contain Any Images", "Demo", MessageBoxButtons.OK, MessageBoxIcon.Asterisk);

}

if (listBox2.Items.Count >= 1)

{

//button13.Enabled = true;

button2.Enabled = true;

}

else

{

button2.Enabled = false;

//button13.Enabled = false;

}

}

}

privatevoidloadImagestoPanel(StringimageName, StringImageFullName, intnewLocX, intnewLocY)

{

PictureBox ctrl = newPictureBox();

ctrl.Image = Image.FromFile(ImageFullName);

ctrl.BackColor = Color.Black;

ctrl.Location = newPoint(newLocX, newLocY);

ctrl.Size = newSystem.Drawing.Size(sizeWidth, sizeHeight);

ctrl.SizeMode = PictureBoxSizeMode.StretchImage;

//ctrl.MouseMove += new MouseEventHandler(control\_MouseMove);

//pnControls.Controls.Add(ctrl);

}

privatevoidcheckforimagesinfolders()

{

inttotalimages = 0;

googleresults.Items.Clear();

for (int i = 1; i <= listBox2.Items.Count; i++)

{

listBox2.SetSelected(i - 1, true);

DirectoryInfodir = newDirectoryInfo((string)listBox2.Text.ToString());

FileInfo[] bmpfiles = dir.GetFiles("\*.jpg");

totalimages = totalimages + bmpfiles.Length;

for (int j = 0; j <= bmpfiles.Length - 1; j++)

{

googleresults.Items.Add(listBox2.Text + "\\" + bmpfiles[j]);

listBox1.Items.Add(bmpfiles[j]);

}

}

ImagesToSearch = newstring[totalimages];

for (int i = 0; i <totalimages; i++)

{

ImagesToSearch[i] = googleresults.Items[i].ToString();

}

textBox18.Text = totalimages.ToString();

}

privatevoidCbir\_Load(object sender, EventArgs e)

{

pictureBox1.Image = bmp;

SourceImage = newBitmap(pictureBox1.Image);

//resultpath = path = Path.GetDirectoryName(Application.ExecutablePath.ToString()) + "\\Result\\";

string path1 = Path.GetDirectoryName(Application.ExecutablePath.ToString()) + "\\Images\\";

string path2 = "";

listBox2.Items.Add(path1.ToString());

checkforimagesinfolders();

DataColumndcItemValue = newDataColumn("Name");

DataColumn dcItemN1 = newDataColumn("Deviation");

dcItemN1.DataType = System.Type.GetType("System.Double");

DataResults.Columns.Add(dcItemValue);

DataResults.Columns.Add(dcItemN1);

}

privatevoid button2\_Click(object sender, EventArgs e)

{

if (pictureBox1.Image == null)

{

MessageBox.Show("Please Select image");

}

else

{

classfication();

}

//dieaseimage();

}

privatevoidclassfication()

{

cmd = newSqlCommand("Truncate table temp1", con);

con.Open();

cmd.ExecuteNonQuery();

con.Close();

CEDD\_Descriptor.CEDDCEDDDescriptor = newCEDD\_Descriptor.CEDD(); // using CEDD.DLL

FCTH\_Descriptor.FCTHFCTHDescriptor = newFCTH\_Descriptor.FCTH(); // using FCTH.DLL

double[] CEDDTable\_Source = newdouble[144];

double[] FCTHTable\_Source = newdouble[192];

double[] CEDDTable\_ToCompare = newdouble[144];

double[] FCTHTable\_ToCompare = newdouble[192];

//Setup Parameters For MPEG-7 Descriptors

// Setup Parameters For MPEG-7 SCD

SCD\_Descriptor Mpeg7SCD = newSCD\_Descriptor(); // using SCD.DLL

double[] SourceSCDTable = newdouble[64];

double[] ToCompareSCDTable = newdouble[64];

// Setup Parameters For MPEG-7 CLD

CLD\_Descriptor Mpeg7CLD = newCLD\_Descriptor(); // using CDL.DLL

int[] SourceYCDL = newint[6];

int[] SourceCbCDL = newint[3];

int[] SourceCrCDL = newint[3];

int[] ToCompareYCDL = newint[6];

int[] ToCompareCbCDL = newint[3];

int[] ToCompareCrCDL = newint[3];

// Setup Parameters For MPEG-7 EHD

EHD\_Descriptor Mpeg7EHD = newEHD\_Descriptor(11); // using EHD.DLL

double[] SourceEHDTable = newdouble[80];

double[] ToCompareEHDTable = newdouble[80];

//Check the descriptor Selected By User

int Descriptor = 0;

/\*

\* 0 for CEDD

\* 1 for FCTH

\* 2 for SCD

\* 3 For CLD

\* 4 For EHD

\* \*/

switch (comboBox1.Text.ToString())

{

case ("CEDD"): Descriptor = 0; break;

case ("FCTH"): Descriptor = 1; break;

case ("SCD"): Descriptor = 2; break;

case ("CLD"): Descriptor = 3; break;

case ("EHD"): Descriptor = 4; break;

default: Descriptor = 0; break;

}

// Get the descriptor for the source image

if (Descriptor == 0)

{

// CEDDTable\_Source = CEDDDescriptor.Apply(SourceImage); // GET The CEDD Descriptor for the Initial Image

Mpeg7CLD.Apply(SourceImage);

for (int h = 0; h < 6; h++)

{

SourceYCDL[h] = Mpeg7CLD.YCoeff[h];

if (h < 3)

{

SourceCbCDL[h] = Mpeg7CLD.CbCoeff[h];

SourceCrCDL[h] = Mpeg7CLD.CrCoeff[h];

// MessageBox.Show(SourceCbCDL[h].ToString());

}

}

}

if (Descriptor == 1) FCTHTable\_Source = FCTHDescriptor.Apply(SourceImage, 2); // GET The FCTH Descriptor for the Initial Image

if (Descriptor == 2)

{

Mpeg7SCD.Apply(SourceImage, 64, 0);

SourceSCDTable = Mpeg7SCD.Norm4BitHistogram;

}

if (Descriptor == 3)

{

Mpeg7CLD.Apply(SourceImage);

for (int h = 0; h < 6; h++)

{

SourceYCDL[h] = Mpeg7CLD.YCoeff[h];

if (h < 3)

{

SourceCbCDL[h] = Mpeg7CLD.CbCoeff[h];

SourceCrCDL[h] = Mpeg7CLD.CrCoeff[h];

}

}

}

if (Descriptor == 4)

{

SourceEHDTable = Mpeg7EHD.Apply(SourceImage);

SourceEHDTable = Mpeg7EHD.Quant(SourceEHDTable); // Dont Forget to Quant The Descriptor

}

// Ok, now you have the descriptor for the images.

// Remember that you can save the descripors for all the images in one XLM type file

// You can use the Img(Rummager) "manage data" to create these files and then search using these files in order to increase the searching speed

intNumberOfImages = Convert.ToInt32(textBox18.Text.ToString());

doubleTotalDeviation = 0; // Here you will store the deviation of the images (pairwise)

BitmapToCompare;

NeurallMPEGComparer = newNeurall(); // Use this class in order to campare images using MPEG-7 Descriptors

DataResults.Clear(); // Clear the results

for (int i = 0; i <NumberOfImages; i++)

{

ToCompare = newBitmap(ImagesToSearch[i].ToString());

if (Descriptor == 0)

{

Mpeg7CLD.Apply(ToCompare);

for (int h = 0; h < 6; h++)

{

ToCompareYCDL[h] = Mpeg7CLD.YCoeff[h];

if (h < 3)

{

ToCompareCbCDL[h] = Mpeg7CLD.CbCoeff[h];

ToCompareCrCDL[h] = Mpeg7CLD.CrCoeff[h];

}

}

TotalDeviation = MPEGComparer.calculateCLDDistance(ToCompareYCDL, ToCompareCbCDL, ToCompareCrCDL, SourceYCDL, SourceCbCDL, SourceCrCDL);

}

if (Descriptor == 1)

{

FCTHTable\_ToCompare = FCTHDescriptor.Apply(ToCompare, 2);

TotalDeviation = TanimotoClassifier(FCTHTable\_ToCompare, FCTHTable\_Source); // We are Using Tanimoto in order to Compare Images

}

if (Descriptor == 2)

{

Mpeg7SCD.Apply(ToCompare, 64, 0);

ToCompareSCDTable = Mpeg7SCD.haarTransformedHistogram;

TotalDeviation = MPEGComparer.calculateSCDDistance(ToCompareSCDTable, SourceSCDTable);

}

if (Descriptor == 3)

{

Mpeg7CLD.Apply(ToCompare);

for (int h = 0; h < 6; h++)

{

ToCompareYCDL[h] = Mpeg7CLD.YCoeff[h];

if (h < 3)

{

ToCompareCbCDL[h] = Mpeg7CLD.CbCoeff[h];

ToCompareCrCDL[h] = Mpeg7CLD.CrCoeff[h];

}

}

TotalDeviation = MPEGComparer.calculateCLDDistance(ToCompareYCDL, ToCompareCbCDL, ToCompareCrCDL, SourceYCDL, SourceCbCDL, SourceCrCDL);

}

if (Descriptor == 4)

{

ToCompareEHDTable = Mpeg7EHD.Apply(ToCompare);

ToCompareEHDTable = Mpeg7EHD.Quant(ToCompareEHDTable);

TotalDeviation = MPEGComparer.calculateEHDDistance(ToCompareEHDTable, SourceEHDTable);

}

pictureBox2.Image = newBitmap(ImagesToSearch[i].ToString());

pictureBox2.SizeMode = PictureBoxSizeMode.StretchImage;

//pictureBox2.Image.Save(path + i.ToString() + ".jpg");

DataResults.Rows.Add(ImagesToSearch[i].ToString(), TotalDeviation);

Doublerr;

rr = Convert.ToDouble(100 - TotalDeviation);

string s = listBox1.Items[i].ToString();

cmd = newSqlCommand("insert into temp1 values ('" + s + "', '" + rr + "','" + ImagesToSearch[i].ToString() + "',@image)", con);

conv\_photo();

con.Open();

cmd.ExecuteNonQuery();

con.Close();

}

callll();

}

stringImageName;

privatevoidcallll()

{

// //output

// cmd = new SqlCommand("SELECT Name,Devation,path FROM temp1 ORDER BY Devation DESC", con);

// SqlDataAdapter da = new SqlDataAdapter(cmd);

// DataTabledt = new DataTable();

// da.Fill(dt);

// // dataGrid4.DataSource = dt;

//// dataGrid4.Refresh();

con.Open();

cmd = newSqlCommand("SELECT \* FROM temp1 ORDER BY Devation DESC", con);

SqlDataReaderdr = cmd.ExecuteReader();

if (dr.Read())

{

//label7.Text = dr["HerbalName"].ToString();

//label8.Text = dr["ScientificName"].ToString();

// label9.Text = dr["Useage"].ToString();

ImageName = dr["path"].ToString();

label11.Text = dr["Devation"].ToString();

}

con.Close();

ImageName = System.IO.Path.GetFileName(ImageName);

con.Open();

cmd = newSqlCommand("SELECT \* FROM regtb where ImageName='" + ImageName + "'", con);

SqlDataReader dr1 = cmd.ExecuteReader();

if (dr1.Read())

{

label7.Text = dr1["Name"].ToString();

// ImageName = dr["Image"].ToString();

}

else

{

}

con.Close();

}

voidconv\_photo()

{

//converting photo to binary data

if (pictureBox1.Image != null)

{

MemoryStreamms = newMemoryStream();

pictureBox1.Image.Save(ms, ImageFormat.Jpeg);

byte[] photo\_aray = newbyte[ms.Length];

ms.Position = 0;

ms.Read(photo\_aray, 0, photo\_aray.Length);

cmd.Parameters.AddWithValue("@image", photo\_aray);

}

}

privatedoubleTanimotoClassifier(double[] Table1, double[] Table2)

{

double Result = 0;

double Temp1 = 0;

double Temp2 = 0;

double TempCount1 = 0, TempCount2 = 0, TempCount3 = 0;

for (int i = 0; i < Table1.Length; i++)

{

Temp1 += Table1[i];

Temp2 += Table2[i];

}

if (Temp1 == 0 || Temp2 == 0) Result = 100;

if (Temp1 == 0 && Temp2 == 0) Result = 0;

if (Temp1 > 0 && Temp2 > 0)

{

for (int i = 0; i < Table1.Length; i++)

{

TempCount1 += (Table1[i] / Temp1) \* (Table2[i] / Temp2);

TempCount2 += (Table2[i] / Temp2) \* (Table2[i] / Temp2);

TempCount3 += (Table1[i] / Temp1) \* (Table1[i] / Temp1);

}

Result = (100 - 100 \* (TempCount1 / (TempCount2 + TempCount3 - TempCount1))); //Tanimoto

}

return (Result);

}

privatevoid button3\_Click(object sender, EventArgs e)

{

OpenFileDialog op = newOpenFileDialog();

if (op.ShowDialog() == DialogResult.OK)

{

this.Cursor = Cursors.WaitCursor;

try

{

SourceImage = newBitmap(op.FileName);

pictureBox1.Image = SourceImage;

}

catch

{

}

this.Cursor = Cursors.Default;

}

}

publicstringiname;

privatevoid button4\_Click\_1(object sender, EventArgs e)

{

}

privatevoid comboBox2\_SelectedIndexChanged(object sender, EventArgs e)

{

}

privatevoid button3\_Click\_1(object sender, EventArgs e)

{

Application.Exit();

}

privatevoid label5\_Click(object sender, EventArgs e)

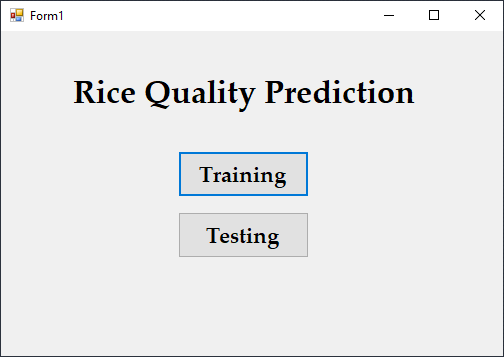
{

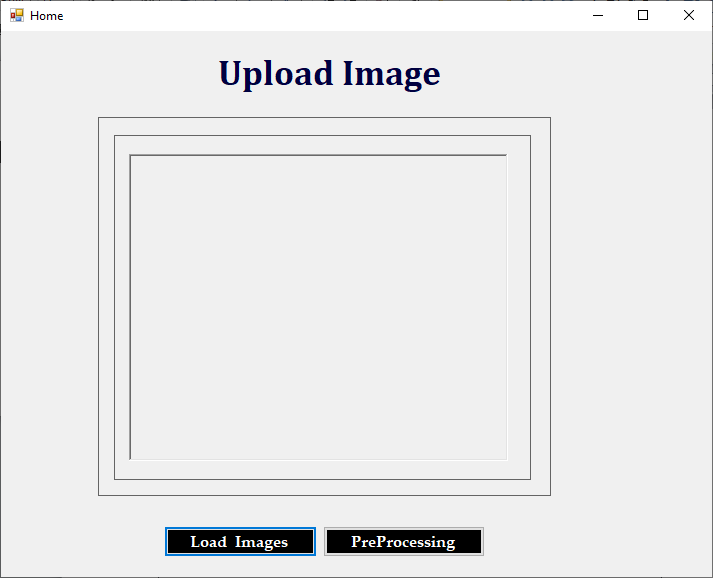
}

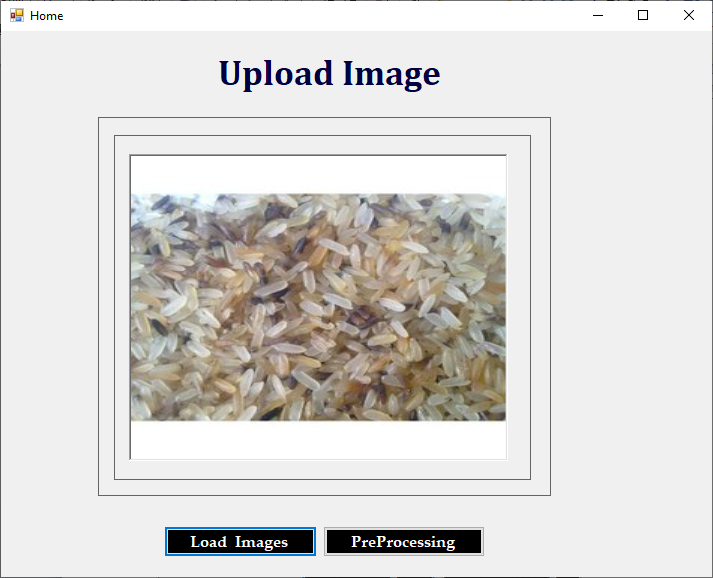
}

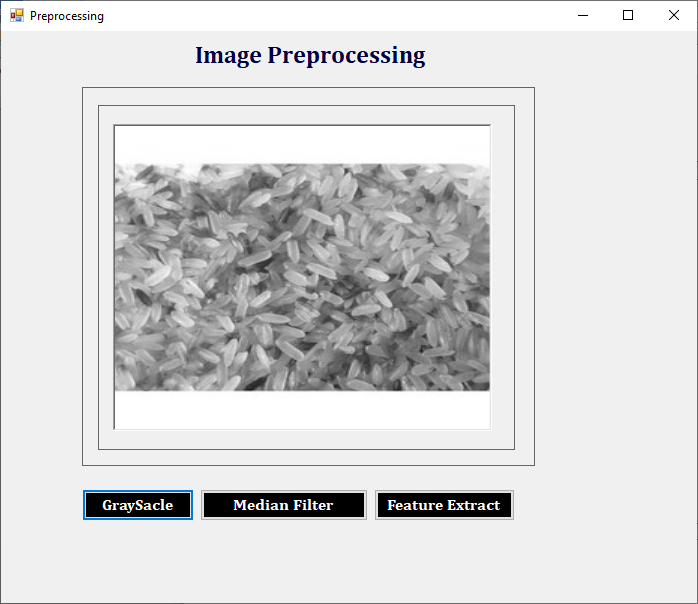
}

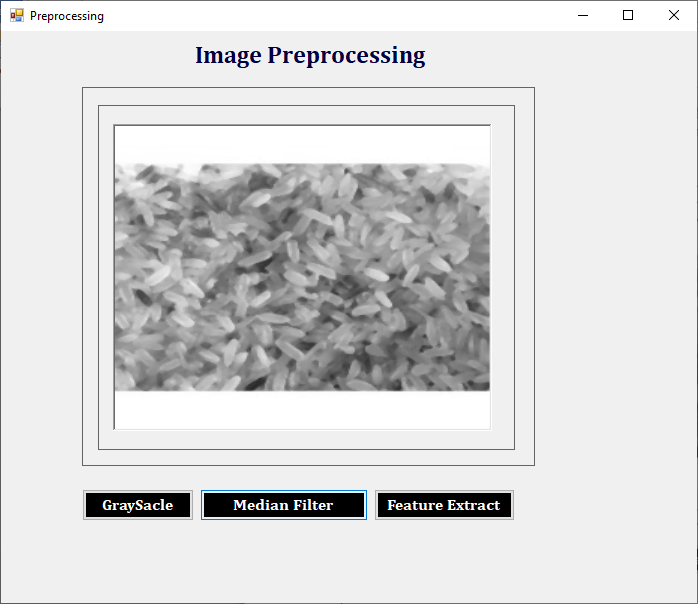
**9.2 SCREENSHOTS**

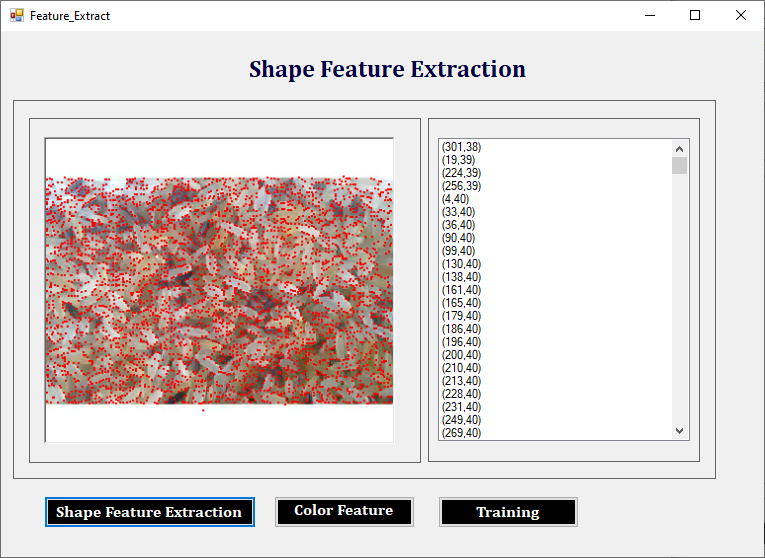


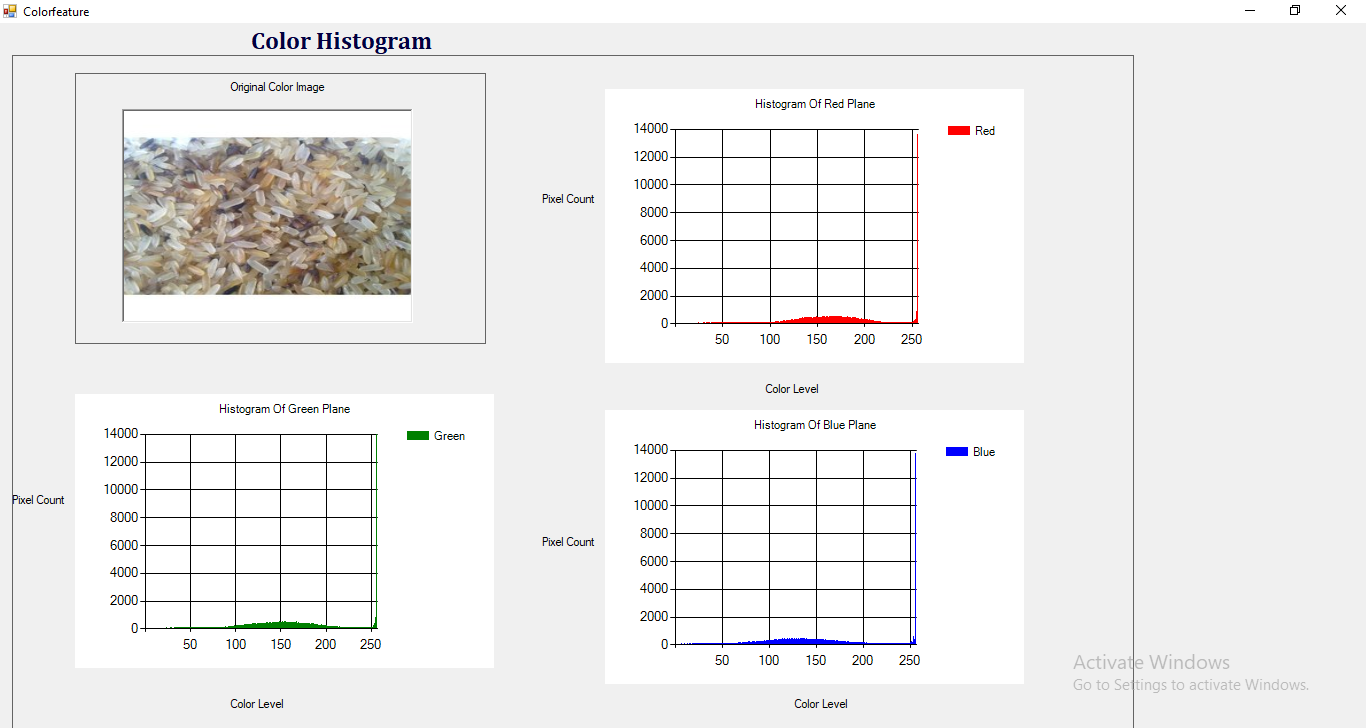


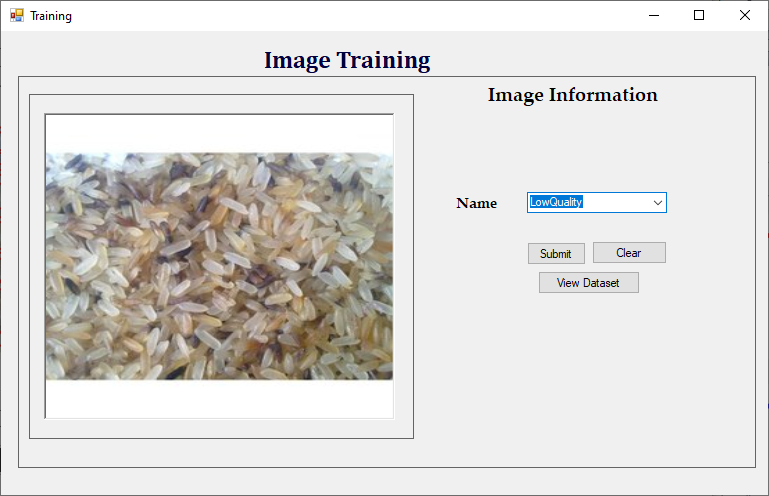


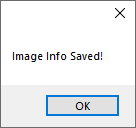


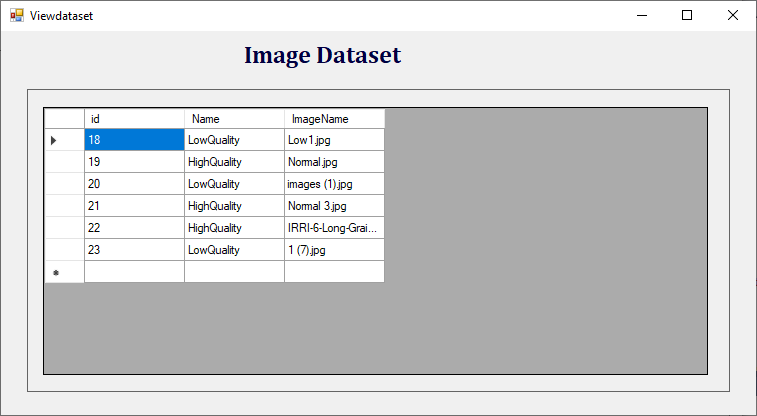


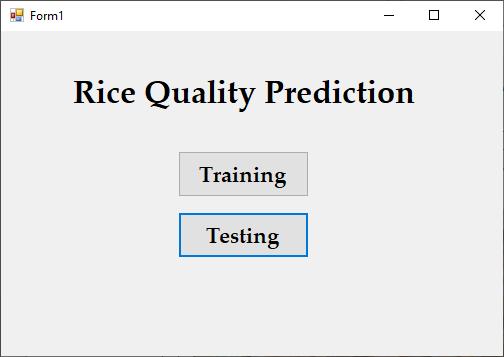


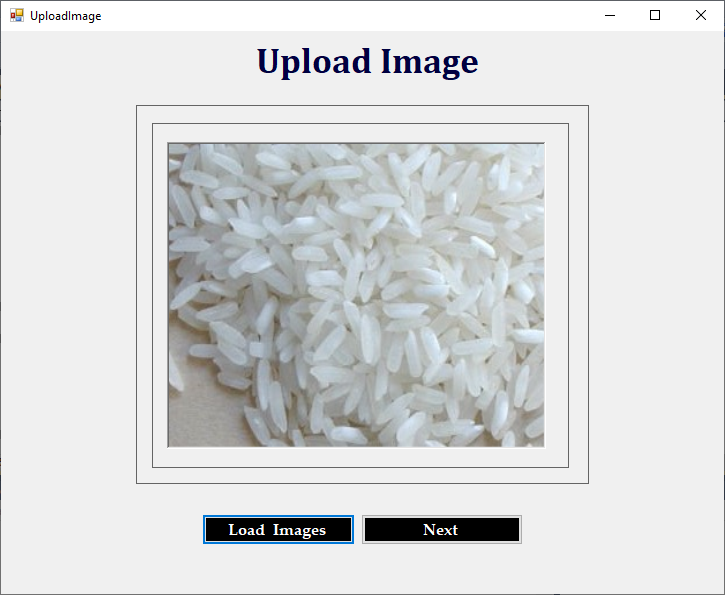






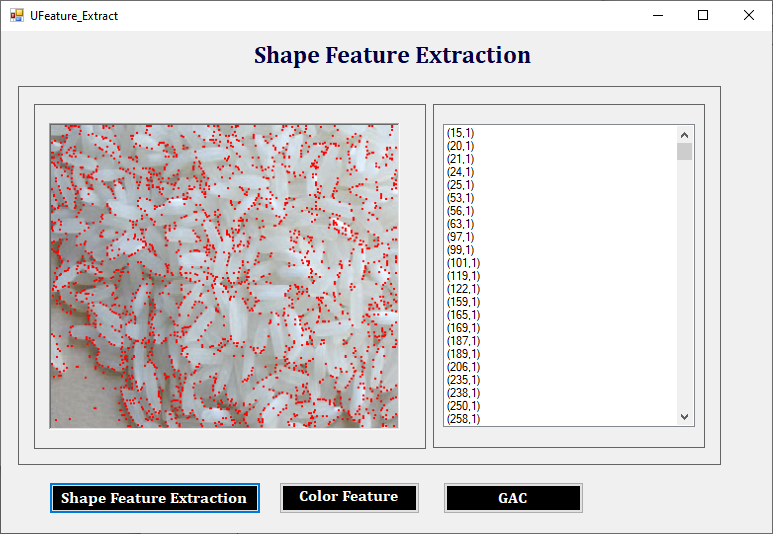


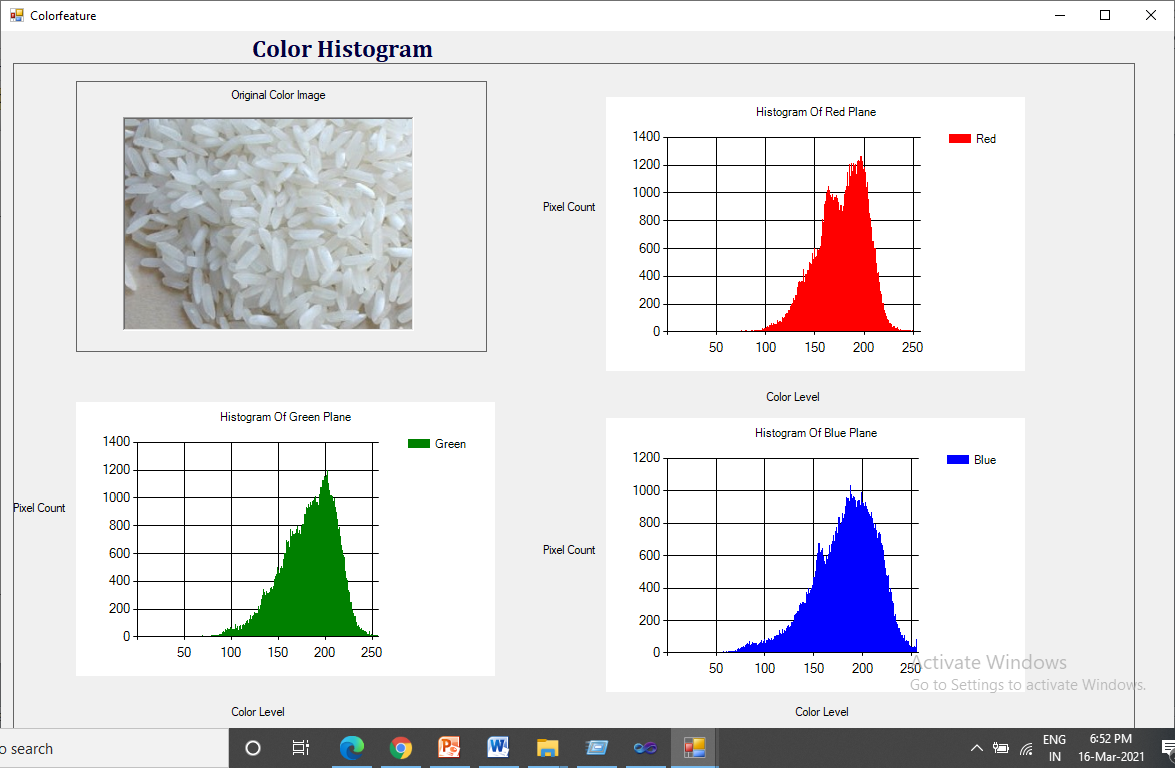


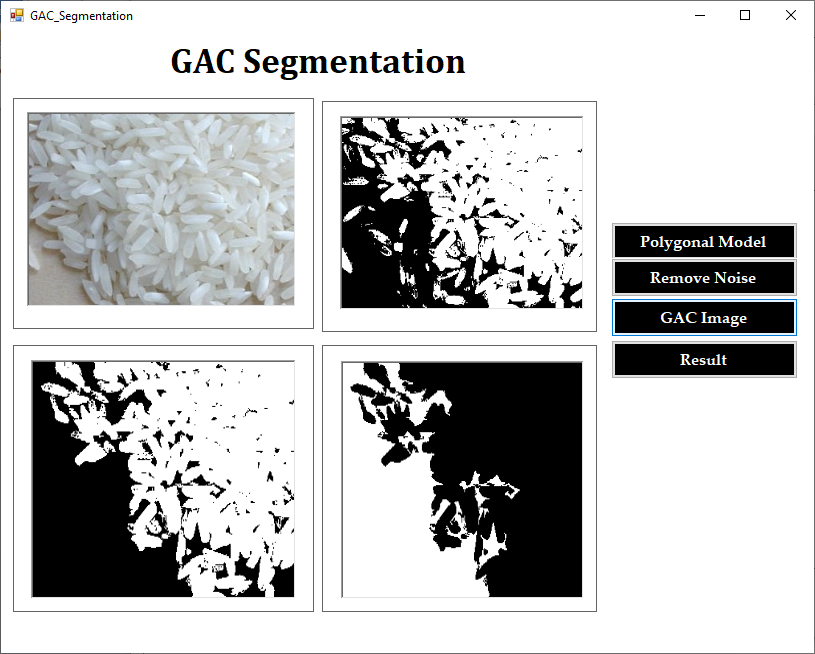


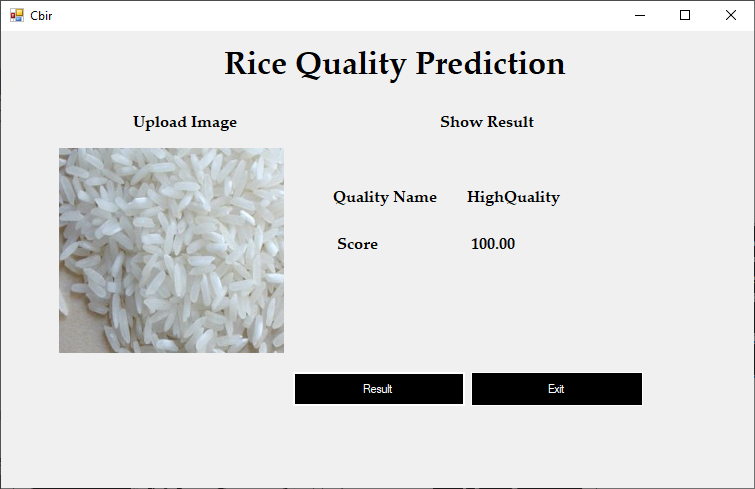


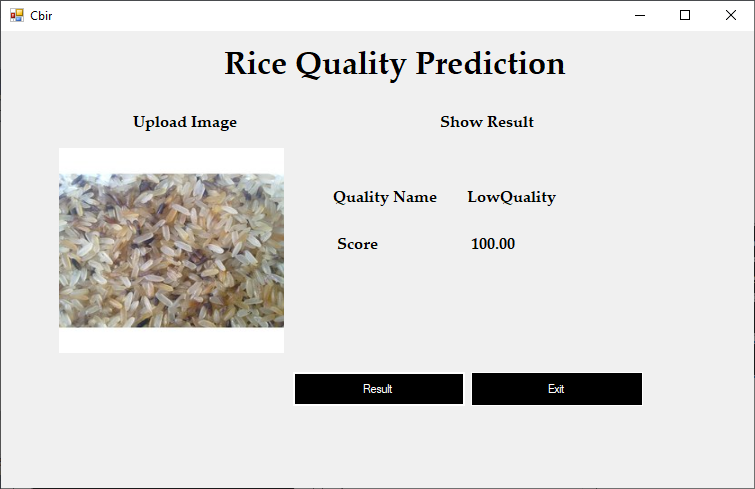












**REFERENCE**

* ASP.NET Core in Action, Second Edition Annotated Edition by Andrew Lock, 2021.
* An Atypical ASP.NET Core 6 Design Patterns Guide: A SOLID adventure into architectural principles and design patterns using .NET 6 and C# 10, 2nd Edition 2nd ed. Edition, 2022, by Carl-Hugo Marcotte (Author), Abdelhamid Zebdi (Foreword)
* ASP.NET Core 5 and React: Full-stack web development using .NET 5, React 17, and TypeScript 4, 2nd Edition 2nd ed. Edition, 2021, by Carl Rippon
* C# 9 and .NET 5 – Modern Cross-Platform Development: Build intelligent apps, websites, and services with Blazor, ASP.NET Core, and Entity Framework Core using Visual Studio Code, 5th Edition 5th ed. Edition, 2020, by Mark J. Price
* Essential ASP.NET Web Forms Development: Full Stack Programming with C#, SQL, Ajax, and JavaScript 1st ed. Edition, 2020, by Robert E. Beasley

**WEBSITE REFERENCE**

* www.csharpcornar.com
* www. Microsoft.com/sql
* www.databasejournal.com
* www.microsoft.com/vcsharp