National College of Ireland

Configuration Manual

MSc Research Project
MSc Data Analytics

Akimuddin Aslam Shaikh
Student ID: x22123245

School of Computing
National College of Ireland

Supervisor: Bharat Agarwal

National College of Ireland
Project Submission Sheet
School of Computing

National College of Ireland

Student Name:	Akimuddin Aslam Shaikh
Student ID:	x22123245
Programme:	MSc Data Analytics
Year:	2024
Module:	MSc Research Project
Supervisor:	Bharat Agarwal
Submission Due Date:	12/12/2024
Project Title:	Configuration Manual
Word Count:	498
Page Count:	10

I hereby certify that the information contained in this (my submission) is information pertaining to research I conducted for this project. All information other than my own contribution will be fully referenced and listed in the relevant bibliography section at the rear of the project.

ALL internet material must be referenced in the bibliography section. Students are required to use the Referencing Standard specified in the report template. To use other author's written or electronic work is illegal (plagiarism) and may result in disciplinary action.

Signature:	Akimuddin Aslam Shaikh
Date:	11th December 2024

PLEASE READ THE FOLLOWING INSTRUCTIONS AND CHECKLIST:

Attach a completed copy of this sheet to each project (including multiple copies).	☐
Attach a Moodle submission receipt of the online project submission, to each project (including multiple copies).	☐
You must ensure that you retain a HARD COPY of the project, both for your own reference and in case a project is lost or mislaid. It is not sufficient to keep a copy on computer.	☐

Assignments that are submitted to the Programme Coordinator office must be placed into the assignment box located outside the office.

Office Use Only	
Signature:	
Date:	
Penalty Applied (if applicable):	

Configuration Manual

Akimuddin Aslam Shaikh
x22123245

1 Introduction

The configuration manual provide details of all the software and hardware that was necessary for the project. It also share the details of the required libraries for models building, importing necessary libraries and all other important aspects of the code that assisted in machine learning computation and visualization.

2 Hardware Specifications

Figure 1 , Figure 2, Figure 3 provides the hardware configuration of laptop device used for the project.

OS Name	Microsoft Windows 10 Home
Version	10.0.19045 Build 19045
Other OS Description	Not Available
OS Manufacturer	Microsoft Corporation
System Name	DESKTOP-NDLVA7S
System Manufacturer	HP
System Model	HP Laptop 15-di2xxx
System Type	x64-based PC
System SKU	8WN04PA#ACJ
Processor	Intel(R) Core(TM) i5-10210U CPU @ 1.60GHz, 2112 Mhz, 4 ...
BIOS Version/Date	Insyde F.41, 4/19/2022
SMBIOS Version	3.2
Embedded Controller V...	88.31
BIOS Mode	UEFI
BaseBoard Manufacturer	HP
BaseBoard Product	86B3
BaseBoard Version	88.31
Platform Role	Mobile
Secure Boot State	Off
PCR7 Configuration	Elevation Required to View
Windows Directory	C:\Windows
System Directory	C:\Windows\system32
Boot Device	\Device\HarddiskVolume3
Locale	United States
Hardware Abstraction L...	Version = "10.0.19041.5072"
User Name	DESKTOP-NDLVA7S\ABCD
Time Zone	GMT Standard Time

Figure 1: System confugration

3 Software and language Used

Software: Google Colab Pro
Language: Python

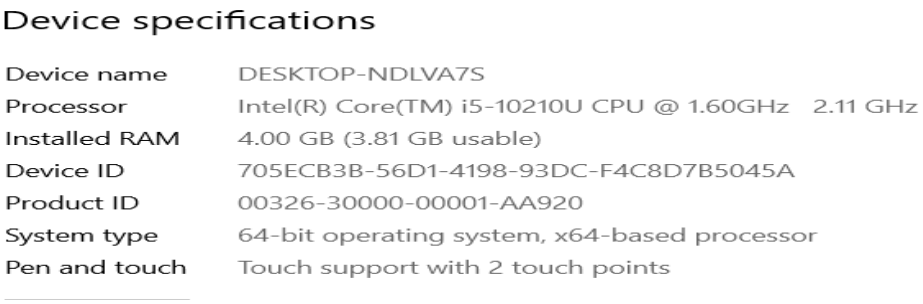

Figure 2: device specification

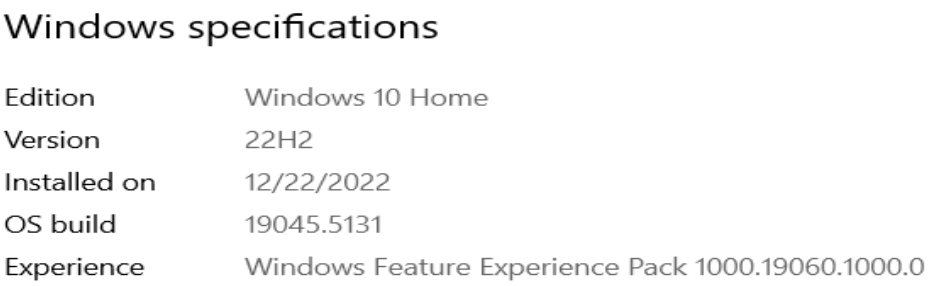

Figure 3: windows specification

4 Python Libraries Used

4.1 File Handling and Data Management

Figure 4 shows us the file management, directories, and data operations Libraries.

```
from google.colab import drive  # For mounting Google Drive in Colab
import zipfile  # For extracting ZIP files
import os  # For file and directory operations
import shutil  # For file and directory management
import hashlib  # For generating file hashes
from collections import defaultdict  # For organizing data structures
```

Figure 4: File Handling and Data Management Libraries

4.2 Data Visualization

Figure 5 depicts libraries to plot visualization thoguh graphs and images.

4.3 Image Processing

Figure 6 demonstrate libraries for image processing and handling image data.

```
import matplotlib.pyplot as plt  # For creating plots and visualizations
import random  # For random sampling
from PIL import Image  # For image handling
```

Figure 5: Data visualization libraries

```
import cv2  # For image processing
import numpy as np  # For numerical operations and arrays
from tensorflow.keras.utils import load_img  # For loading images
from tensorflow.keras.utils import img_to_array  # For converting images to arrays
from skimage.feature import greycomatrix, greycoprops  # For GLCM texture feature extraction
```

Figure 6: Image Processing Libraries

4.4 Machine Learning Utilities

Figure 7 represent tools for machine learning, clustering, dimensionality reduction, and evaluation.

```
from sklearn.model_selection import train_test_split  # For splitting datasets into training and tes
from sklearn.preprocessing import StandardScaler  # For data normalization
from sklearn.metrics import silhouette_score, accuracy_score, classification_report  # For evaluatio
from sklearn.cluster import KMeans, DBSCAN, AgglomerativeClustering  # For clustering techniques
from sklearn.decomposition import PCA  # For dimensionality reduction
from sklearn.manifold import TSNE  # For data visualization
```

Figure 7: Machine Learning Libraries

4.5 Deep Learning

Figure 8 illustrate libraries for pretrained models and utilities for deep learning.

4.6 Classification Models

Figure 9 represent libraries for building and training classification models.

4.7 Progress Monitoring

Figure 10 shows the usage of tqdm to track progress during iterations.

4.8 Data Analysis

Figure 11 represent libraries for analyzing and handling data.

```
from tensorflow.keras.applications import ResNet50  # For feature extraction using ResNet50
from tensorflow.keras.applications.resnet50 import preprocess_input  # For preprocessing images for
```

Figure 8: Deep learning Libraries

```
from sklearn.ensemble import RandomForestClassifier  # For Random Forest classification
from sklearn.svm import SVC  # For Support Vector Machine classification
from xgboost import XGBClassifier  # For XGBoost classification
```

Figure 9: Classification Models Libraries

4.9 External Library Management

Figure 12 shows the libraries to install and manage library versions

5 Dataset

The liver dataset was downloaded from kaggle website. It contain 19261 unlabel image files grouped in subfolders. The overview of the dataset is shown in Figure 13

Dataset Link: https://www.kaggle.com/datasets/anassbenfares/liver-images/data

6 Code implementation snippet

6.1 Data Collection and Extraction

Figure 14 shows the Snippet code for mounting Google Drive and extracting dataset from zipped file.

6.2 Data Preprocessing

Figure 15 provide us with the snippet code for prepocessig all the images into batches for efficient processing and reducing memory overhead.

6.3 Feature Extraction

6.3.1 GLCM Feature

Figure 16 depicts the code for calculating texture-based features using GLCM.

6.3.2 ResNet50 Features

Figure 17 depicts the code for extracting features using a pretrained ResNet50 model.

```
from tqdm import tqdm  # For progress bars during loops
```

Figure 10: Progress Monitoring Libraries

```
import pandas as pd  # For working with tabular data
```

Figure 11: Data Analysis Libraries

6.4 Clustering

6.4.1 KMeans Clustering

Figure 18 shows the Snippet code of applying KMeans for unsupervised clustering.

6.4.2 Agglomerative Clustering

Figure 19 shows the Snippet code of Performing hierarchical clustering.

6.5 Classification

6.5.1 Random forest

Figure 20 shows the Snippet code for training and evaluating a Random Forest classifier and displays classification reports for the same.

6.5.2 XGBoost

Figure 21 shows the Snippet code for training and evaluating an XGBoost classifier and displays classification reports for the same.

6.5.3 Support Vector Machine

Figure 22 shows the Snippet code for training and evaluating an SVM classifier and displays classification reports for the same.

6.6 Data Visualization

6.6.1 Pixel Intensity Analysis

Figure 23 display the Snippet code for plotting the pixel intensity distribution.

6.6.2 PCA Visualization

Figure 24 display the Snippet code for reducing dimensions with PCA and visualizing clusters.

```
!pip install -U scikit-image  # Install the latest version of scikit-image
!pip uninstall scikit-image -y  # Uninstall scikit-image
!pip install scikit-image==0.18.3  # Install a specific version of scikit-image
```

Figure 12: External Libraries

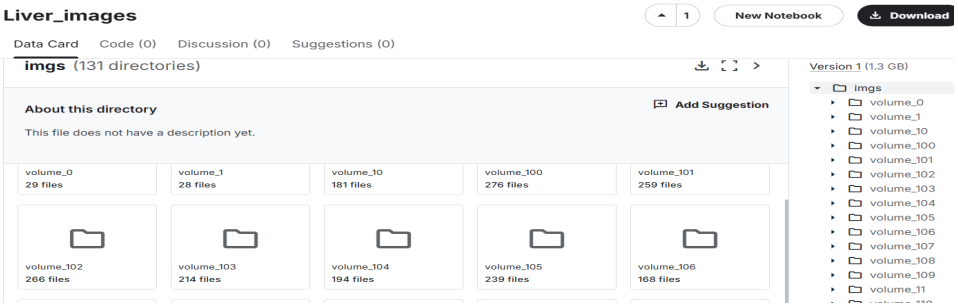

Figure 13: Liver dataset Overview

6.7 Cross Validation

Figure 25 display the code for cross validation to keep consistency for all the stages of performance.

```python
# Path to the zipped file and the extraction folder
zip_file_path = '/content/drive/My Drive/archive (28).zip'
extract_to_path = '/content/drive/My Drive/dataset_folder'

# Check if the dataset is already extracted
if not os.path.exists(extract_to_path):
    print("Extracting dataset...")
    with zipfile.ZipFile(zip_file_path, 'r') as zip_ref:
        zip_ref.extractall(extract_to_path)
    print("Dataset extracted successfully!")
else:
    print("Dataset folder already exists. Skipping extraction.")

# Verify the dataset folder exists
if os.path.exists(extract_to_path):
    print(f"Dataset is ready at: {extract_to_path}")
else:
    print("Error: Dataset folder not found. Please check your paths.")
```

Figure 14: Data Collection and Extraction

```python
# Image processing parameters
image_size = (224, 224)  # Resize all images to 224x224
batch_size = 500  # Number of images per batch

# Step 1: Get all image file paths
all_images = sorted([os.path.join(source_folder, img) for img in os.listdir(source_folder) if img.lower().endswith(('.png', '.jpg', '.jpeg', '.bmp', '.tiff'))])

# Step 2: Check existing .npy files
existing_batches = [file for file in os.listdir(output_dir) if file.startswith('batch_') and file.endswith('.npy')]
existing_batches.sort()  # Sort to maintain order
print(f"Existing .npy files: {len(existing_batches)}")

# Determine already processed images
processed_images = len(existing_batches) * batch_size
if processed_images >= len(all_images):
    print("All images are already processed. No further action required.")
else:
    print(f"Some images are already processed. Resuming from image index {processed_images}...")
```

Figure 15: Data Preprocessing code

```python
# GLCM feature extraction function
def compute_glcm_features(images):
    features = []
    for img in images:
        try:
            # Ensure the image is in grayscale
            if len(img.shape) == 3 and img.shape[-1] == 3:  # Convert RGB to grayscale
                img = np.dot(img[..., :3], [0.2989, 0.5870, 0.1140])
            img = (img * 255).astype(np.uint8)  # Scale to 8-bit for GLCM

            # Compute GLCM
            glcm = greycomatrix(img, distances=[1], angles=[0], levels=256, symmetric=True, normed=True)

            # Compute texture properties
            contrast = greycoprops(glcm, 'contrast')[0, 0]
            dissimilarity = greycoprops(glcm, 'dissimilarity')[0, 0]
            homogeneity = greycoprops(glcm, 'homogeneity')[0, 0]
            energy = greycoprops(glcm, 'energy')[0, 0]
            correlation = greycoprops(glcm, 'correlation')[0, 0]

            # Append features
            features.append([contrast, dissimilarity, homogeneity, energy, correlation])
        except Exception as e:
            print(f"Error processing image: {e}")
            continue
    return np.array(features)
```

Figure 16: GLCM Feature Extraction

```python
# Load the ResNet50 model
resnet_model = ResNet50(weights='imagenet', include_top=False, pooling='avg')  # Global Average Pooling
print("ResNet50 model loaded successfully.")
```

```
Downloading data from https://storage.googleapis.com/tensorflow/keras-applications/resnet/resnet50_weights_tf_dim_ordering_tf_kernels_notop.h5
94765736/94765736 ━━━━━━━━━━━━━━━━━━━━ 0s 0us/step
ResNet50 model loaded successfully.
```

```python
def extract_features(image_paths, batch_size=100, image_size=(224, 224)):
    features = []
    for i in tqdm(range(0, len(image_paths), batch_size)):
        batch_paths = image_paths[i:i + batch_size]
        batch_images = []

        for path in batch_paths:
            # Load and preprocess each image
            img = load_img(path, target_size=image_size)
            img_array = img_to_array(img)
            img_array = preprocess_input(img_array)  # Preprocess for ResNet50
            batch_images.append(img_array)

        # Convert to numpy array and extract features
        batch_images = np.array(batch_images)
        batch_features = resnet_model.predict(batch_images)
        features.append(batch_features)

    return np.vstack(features)
```

Figure 17: ResNet50 Features

```python
# from sklearn.cluster import KMeans

# Flatten images for clustering
flat_data = combined_data.reshape(combined_data.shape[0], -1)  # Shape: (19261, 224*224*3)

# Apply KMeans clustering
num_clusters = 5  # Set the number of clusters
kmeans = KMeans(n_clusters=num_clusters, random_state=42)
labels = kmeans.fit_predict(flat_data)

print(f"Cluster labels for all images: {labels}")
```

Figure 18: KMeans Clustering

```python
# from sklearn.cluster import AgglomerativeClustering

agglo = AgglomerativeClustering(n_clusters=5)  # Try with 5 clusters
agglo_labels = agglo.fit_predict(reduced_features)

num_clusters = len(set(agglo_labels))
print(f"Number of clusters: {num_clusters}")
```

Figure 19: Agglomerative Clustering

```python
# from sklearn.ensemble import RandomForestClassifier
# from sklearn.metrics import accuracy_score, classification_report

# Train a Random Forest model
rf_model = RandomForestClassifier(n_estimators=100, random_state=42)
rf_model.fit(X_train, y_train)

# Evaluate the model
y_pred = rf_model.predict(X_test)
accuracy = accuracy_score(y_test, y_pred)
print(f"Random Forest Accuracy: {accuracy:.2f}")
print(classification_report(y_test, y_pred))
```

Figure 20: Random forest

```python
# from xgboost import XGBClassifier

# Train an XGBoost model
xgb_model = XGBClassifier(use_label_encoder=False, eval_metric='logloss', random_state=42)
xgb_model.fit(X_train, y_train)

# Evaluate the model
y_pred_xgb = xgb_model.predict(X_test)
accuracy_xgb = accuracy_score(y_test, y_pred_xgb)
print(f"XGBoost Accuracy: {accuracy_xgb:.2f}")
print(classification_report(y_test, y_pred_xgb))
```

Figure 21: XGBoost

```python
# from sklearn.svm import SVC
# Initialize and train SVM
svm_model = SVC(kernel='rbf', class_weight='balanced', random_state=42)

svm_model.fit(X_train, y_train)

# Predict using SVM
y_pred_svm = svm_model.predict(X_test)
accuracy_svm = accuracy_score(y_test, y_pred_svm)
print(f"SVM Accuracy: {accuracy_svm:.2f}")
print("Classification Report for SVM:")
print(classification_report(y_test, y_pred_svm))
```

Figure 22: SVM

```python
# Select 100 random images for pixel intensity analysis
sample_images = random.sample(all_images, 100)

# Gather pixel intensities
pixel_values = []
for img_path in sample_images:
    img = load_img(img_path, target_size=(224, 224))  # Resize for consistency
    img_array = img_to_array(img) / 255.0  # Normalize pixel values
    pixel_values.extend(img_array.flatten())  # Flatten the array and collect values

# Plot histogram
plt.figure(figsize=(10, 5))
plt.hist(pixel_values, bins=50, color='blue', alpha=0.7)
plt.title("Pixel Intensity Distribution")
plt.xlabel("Pixel Value")
plt.ylabel("Frequency")
plt.show()
```

Figure 23: Pixel Intensity distribution code

```python
# from sklearn.decomposition import PCA

# Reduce dimensionality for visualization
pca = PCA(n_components=2)
reduced_data = pca.fit_transform(flat_data)

# Plot the reduced data
plt.figure(figsize=(10, 7))
for cluster in unique:
    cluster_points = reduced_data[labels == cluster]
    plt.scatter(cluster_points[:, 0], cluster_points[:, 1], label=f"Cluster {cluster}", alpha=0.6)
plt.legend()
plt.title("2D PCA Visualization of Clusters")
plt.show()
```

Figure 24: PCA Visualization code

```python
#ResNet + KMeans labels
kmeans_labels = cluster_labels
X_train, X_test, y_train, y_test = train_test_split(normalized_combined_features, kmeans_labels, test_size=0.2, random_state=42)

#ResNet + Agglomerative labels
X_train, X_test, y_train, y_test = train_test_split(normalized_combined_features, agglo_labels, test_size=0.2, random_state=42)

#GLCM + Agglomerative labels
X_train, X_test, y_train, y_test = train_test_split(combined_features, agglo_labels, test_size=0.2, random_state=42)

#GLCM +kMEANS labels
X_train, X_test, y_train, y_test = train_test_split(combined_features, kmeans_labels, test_size=0.2, random_state=42)

#(GLCM + ResNet) + Kmeans labels
X_train, X_test, y_train, y_test = train_test_split(combined_features, kmeans_labels, test_size=0.2, random_state=42)

#(GLCM + ResNet) + Agglomerative
X_train, X_test, y_train, y_test = train_test_split(combined_features, agglo_labels, test_size=0.2, random_state=42)
```

Figure 25: Cross validation