Configuration Manual

MSc Research Project
MSCAI1

Meenu Mohan
Student ID : x23119390

School of Computing
National College of Ireland

Supervisor: Kislay Raj

National College of Ireland
Project Submission Sheet
School of Computing

Student Name:	Meenu Mohan
Student ID:	x23119390
Programme:	MSCAI1
Year:	2024
Module:	MSc Research Project
Supervisor:	Kislay Raj
Submission Due Date:	12/08/2024
Project Title:	Configuration Manual
Word Count:	479
Page Count:	8

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:	Meenu Mohan
Date:	11th August 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

Meenu Mohan
x23119390

1 System Configuration

The project is done on a 64-bit Windows 11 operating system with 16GB RAM with AMD Ryzen 7 5800H processor has a base clock speed of 3.20 GHz.

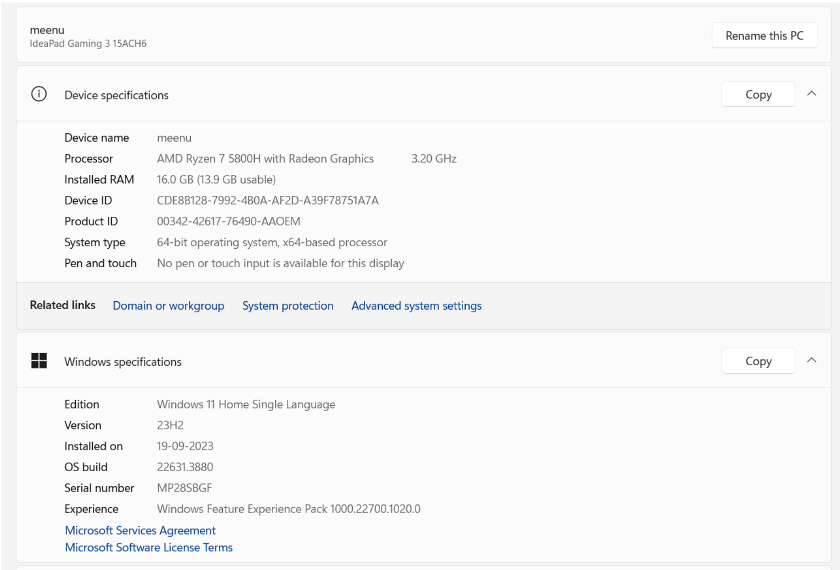

Figure 1: System Configuration

2 Software Requirement

For the project we have used the following software :

1. Python 3.11.4
2. conda 23.7.2
3. Visual Studio Code

Version: 1.92.0 (user setup)
Commit: b1c0a14de1414fcdaa400695b4db1c0799bc3124
Date: 2024-07-31T23:26:45.634Z
Electron: 30.1.2

ElectronBuildId: 9870757
Chromium: 124.0.6367.243
V8: 12.4.254.20-electron.0
OS: Windows$_NT x$6410.0.22631

3 Python Libraries

1. Tensorflow
2. Keras
3. Matplotlib
4. Shap
5. Lime
6. DecisionTreeClassifier
7. PIL
8. numpy
9. cv2
10. scipy
11. skimage

4 Datasets

Selected two datasets from Kaggle for chest cancer and COVID-19 detection both created by Mohamed Hany and Paul Mooney respectively.

Chest Cancer Dataset Link: https://www.kaggle.com/datasets/mohamedhanyyy/chest-ctscan-images/data
Covid Dataset link: https://www.kaggle.com/datasets/paultimothymooney/chest-xray-pneumonia
Both of these datasets contain images of CT scans and an X-ray of the chest to find the disease.

Figure 2: CT scan image of the chest to detect cancer

Figure 3: X-ray image of the chest to detect covid

5 Data Preprocessing

5.1 Combining Dataset

Used two datasets and checked both datasets in the code. Created models for both cancer and COVID-19.

```
def predict_image_quality(img_path):
        # Check file size first
        if file_size < 15:
            return 'low'
        else:
            return 'high'

        img_array = preprocess_image(img_path)
        if img_array is None:
            return 'low'  # Return 'low' if the image cannot be preprocessed

        # Convert image to grayscale
        gray = cv2.cvtColor(img_array[0], cv2.COLOR_BGR2GRAY)

        # Check image sharpness
        sharpness = cv2.Laplacian(gray, cv2.CV_64F).var()

        # Check image contrast
        contrast = gray.std()

        # Define thresholds
        sharpness_threshold = 50
        contrast_threshold = 50

        # Determine quality
        if sharpness > sharpness_threshold and contrast > contrast_threshold:
            return 'high'
        else:
            return 'low'
    except Exception as e:
        print(f"Warning: Unable to determine quality for {img_path}. Error: {e}")
        return 'unknown'  # Return 'unknown' as default if an exception occurs
```

Figure 4: Processing step 1

5.2 Cleaning and organizing image data

```
def is_image_file(filename):
    image_extensions = ['.jpg', '.jpeg', '.png', '.bmp', '.tif', '.tiff']
    return any(filename.lower().endswith(ext) for ext in image_extensions)
```

Figure 5: Processing step 2

5.3 Preprocessing images for model input

Preprocessing the image here in this code to check the quality of the uploaded image.

```
def preprocess_image(img_path, target_size=(224, 224)):
    try:
        img = image.load_img(img_path, target_size=target_size)
        img_array = image.img_to_array(img)
        img_array = np.expand_dims(img_array, axis=0)
        img_array /= 255.0
        return img_array
    except UnidentifiedImageError:
        print(f"Warning: Unable to open image file {img_path}. Skipping.")
        return None
```

Figure 6: Processing step 3

6 Data Analysis

6.1 Examining Dataset Distribution

analyzes the dataset structure and counts images in each class.

```
def analyze_dataset(base_dir):
    for disease_type in ['cancer', 'covid']:
        for split in ['train', 'test', 'valid']:
            path = os.path.join(base_dir, disease_type, split)
            classes = os.listdir(path)
            for cls in classes:
                class_path = os.path.join(path, cls)
                num_images = len([f for f in os.listdir(class_path) if is_image_file(f)])
                print(f"{disease_type} - {split} - {cls}: {num_images} images")
```

Figure 7: Dataset Distribution

6.2 Disease Classification Models

This function loads pre-trained models for cancer and COVID-19 detection.

```
def load_models():
    cancer_model = load_model('cancer_model.h5')
    covid_model = load_model('covid_model.h5')

    return cancer_model, covid_model
```

Figure 8: Classification Models

6.3 Image Enhancement Techniques

These functions enhance image sharpness and contrast, apply super-resolution, and denoise images.

```python
def enhance_image(img_array):
    # Convert to PIL Image
    img = Image.fromarray((img_array[0] * 255).astype(np.uint8))

    # Enhance sharpness
    enhancer = ImageEnhance.Sharpness(img)
    img = enhancer.enhance(2.0)

    # Enhance contrast
    enhancer = ImageEnhance.Contrast(img)
    img = enhancer.enhance(1.5)

    # Convert back to numpy array
    enhanced_array = np.array(img) / 255.0
    return np.expand_dims(enhanced_array, axis=0)

def apply_super_resolution(img_array):
    # Simple bicubic interpolation for super-resolution
    img = img_array[0]
    img = cv2.resize(img, (448, 448), interpolation=cv2.INTER_CUBIC)
    img = cv2.resize(img, (224, 224), interpolation=cv2.INTER_AREA)
    return np.expand_dims(img, axis=0)

def denoise_image(img_array):
    # Apply median filter for denoising
    denoised = ndimage.median_filter(img_array[0], size=3)
    return np.expand_dims(denoised, axis=0)
```

Figure 9: Enhancement Techniques

7 Explainable AI Techniques

These functions implement Grad-CAM, SHAP, and LIME for model interpretability.

```python
if original_quality == 'low':
    print("Generating Grad-CAM visualization...")
    visualize_grad_cam(img_array, model)

    print("Generating SHAP explanation...")
    explain_prediction_with_shap(img_array, model)

    print("Generating LIME explanation...")
    explain_prediction_with_lime(img_array, model, classes)
    enhanced_quality = predict_image_quality(enhanced_img_path)
```

Figure 10: Code for generation SHAP and LIME

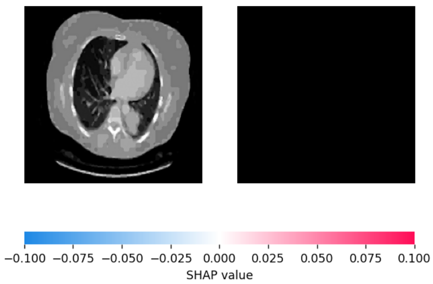

Figure 11: SHAP Value

8 Single Image Prediction

This function predicts and analyzes a single image, handling both high and low-quality images.

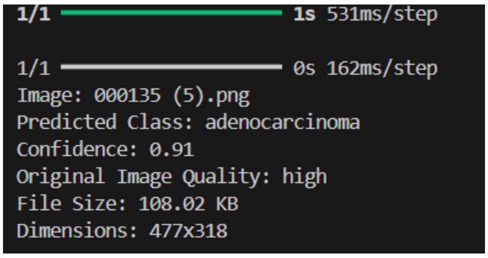

Figure 12: Output of single image prediction

Figure 13: To predict single image

8.1 Handling both high and low-quality images

The project will check the quality of the project as low and high. Here is a sample of low-quality and high-quality images.

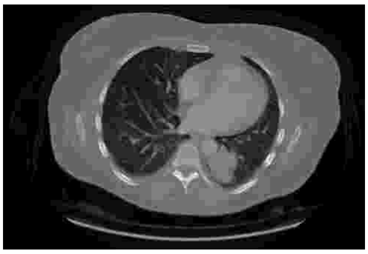

Figure 14: Low-quality image of chest

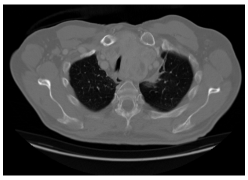

Figure 15: High-quality image of chest

8.2 Applying enhancement techniques for low-quality images

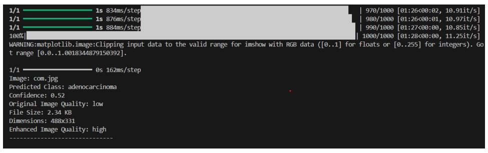

Figure 16: Applying enhancement technique to make low to high

9 Multiple Image Prediction

This function predicts and analyzes multiple images, handling both cancer and COVID-19 datasets.

9.1 Implementing function to predict multiple images

Figure 17: Code for multiple images

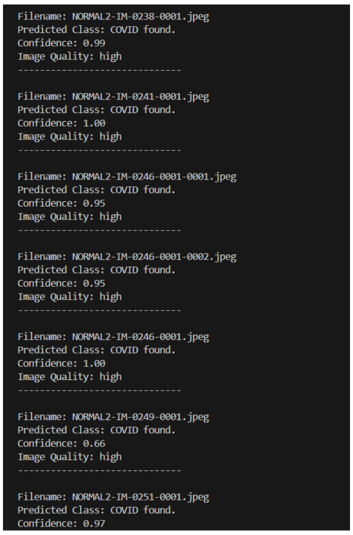

Figure 18: Output of multiple images

10 Decision Tree Algorithm

Applied decision tree to check the accuracy after checking multiple images.

Overall Accuracy: 0.8301

	precision	recall	f1-score	support
COVID	0.79	0.99	0.88	1487
NORMAL	0.96	0.57	0.72	891
accuracy			0.83	2378
accuracy			0.83	2378
accuracy			0.83	2378
macro avg	0.88	0.78	0.80	2378
macro avg	0.88	0.78	0.80	2378
weighted avg	0.86	0.83	0.82	2378

Figure 19: Decision tree algorithm