National College of Ireland

Harnessing AI for Accurate Diabetic Retinopathy Detection in Retinal Imaging

Research Project/Practicum (MSCAI1)
Master of Science in Artificial Intelligence

Muhammad Adeel Iqbal
Student ID: 23380225

School of Computing
National College of Ireland

Supervisor: Dr. Abdul Shahid

National College of Ireland
Project Submission Sheet
School of Computing

Student Name:	Muhammad Adeel Iqbal
Student ID:	23380225
Programme:	Master of Science in Artificial Intelligence
Year:	2024-2025
Module:	Research Project/Practicum (MSCAI1)
Supervisor:	Dr. Abdul Shahid
Submission Due Date:	01/09/2025
Project Title:	Harnessing AI for Accurate Diabetic Retinopathy Detection in Retinal Imaging
Word Count:	1120
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:	Muhammad Adeel Iqbal
Date:	24th August 2025

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):	

Harnessing AI for Accurate Diabetic Retinopathy Detection in Retinal Imaging

Muhammad Adeel Iqbal
23380225

1 Introduction

1.1 Purpose of the Configuration Manual

This configuration manual provides a step-by-step reference for setting up, configuring, and running the unified deep learning framework for early-stage Diabetic Retinopathy (DR) detection and lesion-level segmentation. It guarantees the reproducibility of the environment and enables researchers, practitioners, and system administrators to install the necessary dependencies, run the model training and evaluation process, and encounter minimal technical obstacles. The document can also be used as a reference to have uniformity in deployments across the various machines.

1.2 System Overview

The system developed is a hybrid deep learning pipeline with a classification module (DR presence) and a segmentation module (microaneurysms and Related Lesions Localization) into one. It runs on high-res images obtained by the e-ophtha MA retinal fundus dataset under a streamlined process by combining preprocessing, augmentation, and inference. It performs best in GPU-accelerated settings and features a ResNet34-based U-Net++ backbone as part of the segmentation and a light CNN classifier (Zhou et al. 2018, Koonce 2021). The resulting configuration is well-suited to research and clinical use in screening procedures due to high accuracy, fast inference, and easy-to-interpret lesion probability maps.

2 System Requirements

2.1 Hardware Requirements

2.1.1 Local Machine Configuration

The development initially began on a local machine without dedicated GPU acceleration. The following are the specifications of the system on which I have worked;

- **CPU:** Core(TM) i7-11390H @ 3.40 GHz
- **RAM:** 16.0 GB
- **Disk Space:** 477 GB

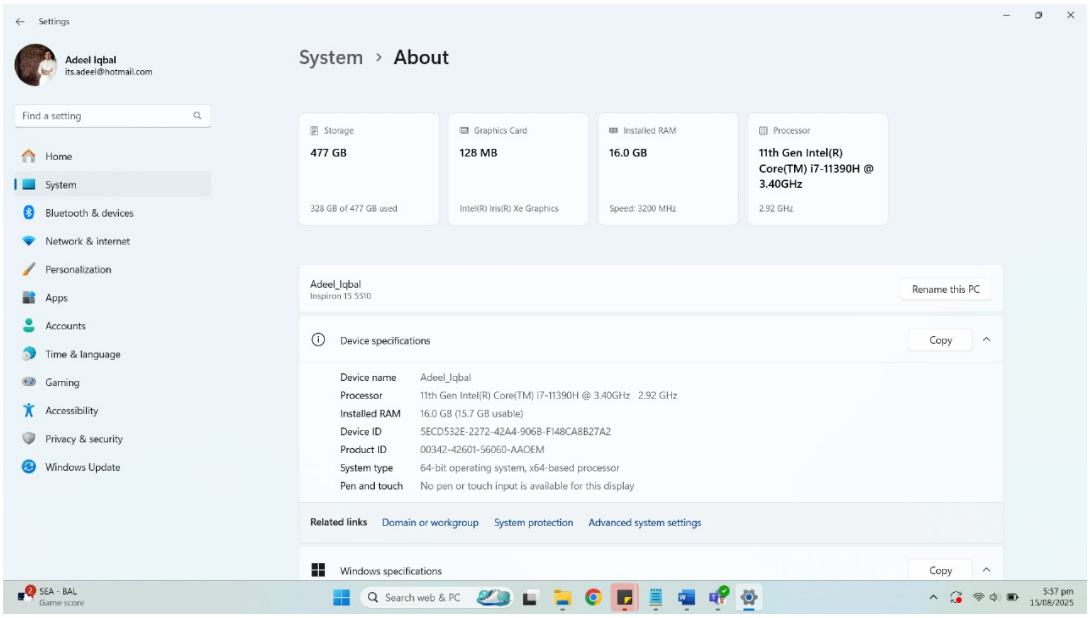

Figure 1: System Information

2.1.2 GPU Configuration (Kaggle Cloud Environment)

Kaggle's GPU-accelerated environment was employed to execute training and inference efficiently because of the unavailability of a GPU on the local machine. The allocated session hardware resources were:

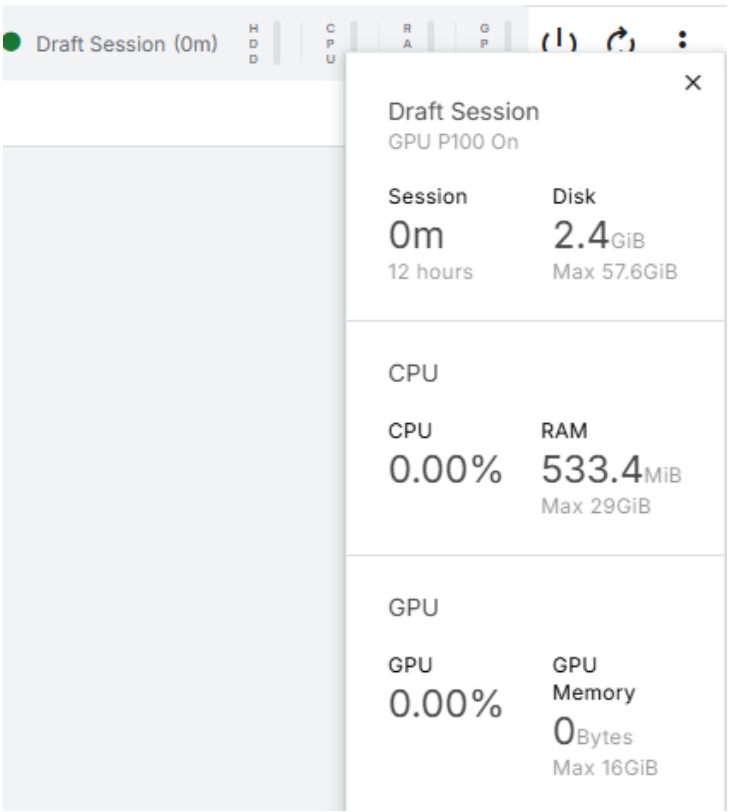

Figure 2: GPU configuration in the Kaggle environment showing Tesla P100 with 16 GB memory.

- **GPU Model**: NVIDIA Tesla P100-PCIE-16GB
- **GPU Capability**: 6.0
- **GPU Total Memory**: 15.89 GB
- **CUDA Version (from PyTorch)**: 12.4
- **cuDNN Version**: 90300
- **Disk Space**: 2.2 GiB used / 57.6 GiB available
- **RAM Usage**: 29 GiB maximum

2.2 Software Requirements

2.2.1 Operating System

Windows 11 Home Single Language, Version 23H2

2.2.2 Python

- Python 3.11.11

2.2.3 Libraries and Frameworks

The following libraries and frameworks were utilized:

- NumPy: 1.26.4
- Pillow: 11.1.0
- PyTorch: 2.6.0+cu124
- TorchVision: 0.21.0+cu124
- scikit-learn: 1.2.2
- Matplotlib: 3.7.2
- tqdm: 4.67.1

2.2.4 Tools

- Kaggle Notebook Environment
- Jupyter Interface

2.3 Network Requirements

A stable broadband internet connection ($\geq$5 Mbps) was required for the download of the dataset, the training of the model, and the execution of the notebook in Kaggle.

3 Installation Instructions

3.1 Prerequisites

Before initiating the installation, ensure the following prerequisites are satisfied:

- A registered Kaggle account with access to GPU-accelerated notebooks.
- Stable broadband internet connection for downloading datasets and dependencies.
- Availability of the required datasets within the Kaggle environment that is uploaded by the user.

3.2 Installation Steps

The installation process is executed within Kaggle's cloud environment. The steps undertaken were:

1. **Dataset Download:** Access the official E-ophtha portal at `https://www.adcis.net/en/third-party/e-ophtha/`, complete the download request form, and obtain the dataset: *e-ophtha_MA* (microaneurysms).
2. **Setup Kaggle:** Log in to Kaggle, create a new notebook, and enable GPU acceleration (select NVIDIA Tesla P100).
3. **Upload Notebook:** Import the Jupyter Notebook (.ipynb file) into the Kaggle workspace to execute the DR detection pipeline.
4. **Upload the Dataset:** Add the downloaded zip file (*e-ophtha_MA.zip* to Kaggle directly via file upload in the notebook environment.
5. **Install Required Libraries:** Install or verify required Python packages using Kaggle's terminal or notebook cells. The following commands ensure all necessary dependencies are available (Culjak 2012, Buslaev et al. 2020, Kramer 2016, Marcel & Rodriguez 2010) :

```
# Core ML/DL Frameworks
pip install torch torchvision torchaudio --upgrade

# Numerical and scientific computing
pip install numpy==1.26.4
pip install scikit-learn==1.2.2
pip install matplotlib==3.7.2
pip install tqdm==4.67.1
pip install pillow==11.1.0

# Deep learning segmentation models
pip install segmentation-models-pytorch

# Image processing
pip install opencv-python
```

```
pip install albumentations

# Utilities
pip install pandas
pip install seaborn
```

Kaggle typically provides CUDA-enabled PyTorch and most core libraries by default. However, executing the above ensures compatibility with the project code.

6. **Run the Code:** Execute the notebook sequentially. This includes preprocessing, model training, evaluation, and inference. Monitor outputs and visualizations directly via notebook cells.

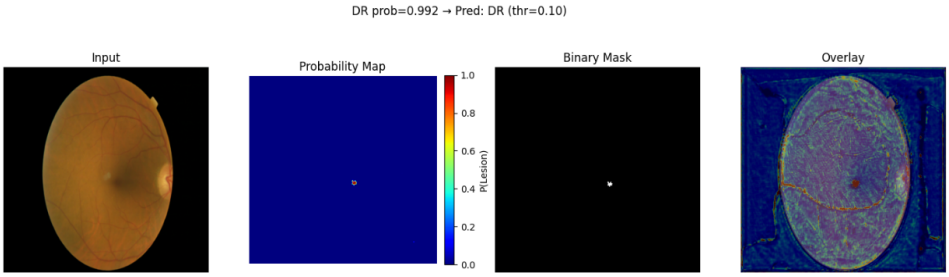

Figure 3: Sample output visualization.

The figure above shows segmentation masks overlaid on retinal images.

4 Conclusion

This configuration manual provided a structured overview of the environment setup, hardware and software requirements, and step-by-step installation process necessary for implementing the proposed system. The hardware setup was based on a Tesla P100 GPU hosted on Kaggle, which facilitated the training of the models in the most efficient way when there is no local GPU available. Specifications of Python version, required libraries, the support of CUDA, and access to GPU memory were described in detail to provide reproducibility of experiments in a similar environment.

In the system configuration, Python 3.11.11 as well as core packages like NumPy, Pillow, PyTorch, TorchVision, scikit-learn, Matplotlib, and tqdm were also installed. These libraries were used as a basis for data preprocessing, deep learning models development, and evaluation. Support of CUDA and cuDNN further guaranteed that the framework was able to take advantage of GPU acceleration to carry out computations on offensive input faster. Clear installation guides were also included in the manual, which included downloading and installing the dataset and setting up the Kaggle environment, along with references to installing the libraries and running the notebook.

The configuration and installation instructions are given, allowing one to repeat the experimental setting in multiple sessions in a unified way. This allows for reproducing and extending the classification-segmentation scheme of medical image analysis without controversy. The manual is consequently considered a guide to prepare and implement the system, which gives transparency and consistency to the researchers and practitioners wishing to follow a similar workflow.

References

Buslaev, A., Iglovikov, V. I., Khvedchenya, E., Parinov, A., Druzhinin, M. & Kalinin, A. A. (2020), 'Albumentations: Fast and flexible image augmentations', *Information 2020, Vol. 11, Page 125* **11**, 125.
URL: *https://www.mdpi.com/2078-2489/11/2/125/htm https://www.mdpi.com/2078-2489/11/2/125*

Culjak, I. (2012), 'A brief introduction to opencv — ieee conference publication — ieee xplore'.
URL: *https://ieeexplore.ieee.org/abstract/document/6240859*

Koonce, B. (2021), 'Resnet 34', *Convolutional Neural Networks with Swift for Tensorflow* pp. 51–61.
URL: *https://link.springer.com/chapter/10.1007/978-1-4842-6168-2*$_5$

Kramer, O. (2016), 'Scikit-learn', *Studies in Big Data* **20**, 45–53.
URL: *https://link.springer.com/chapter/10.1007/978-3-319-33383-0*$_5$

Marcel, S. & Rodriguez, Y. (2010), 'Torchvision the machine-vision package of torch', *MM'10 - Proceedings of the ACM Multimedia 2010 International Conference* pp. 1485–1488.
URL: */doi/pdf/10.1145/1873951.1874254?download=true*

Zhou, Z., Siddiquee, M. M. R., Tajbakhsh, N. & Liang, J. (2018), 'Unet++: A nested u-net architecture for medical image segmentation', *Lecture Notes in Computer Science (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics)* **11045 LNCS**, 3–11.
URL: *https://link.springer.com/chapter/10.1007/978-3-030-00889-5*$_1$