Deep Learning-Based Image Segmentation for Dermatological Lesions

BY

LIM JIA HONG

A REPORT

SUBMITTED TO

Universiti Tunku Abdul Rahman

in partial fulfillment of the requirements

for the degree of

BACHELOR OF INFORMATION TECHNOLOGY (HONOURS) COMMUNICATIONS AND NETWORKING

Faculty of Information and Communication Technology

(Kampar Campus)

JUNE 2025

COPYRIGHT STATEMENT

ACKNOWLEDGEMENTS

I would like to express my sincere thanks and appreciation to my supervisors, Ms Oh Zi Xin who has given me this bright opportunity to engage in a deep learning segmentation project. It is my first step to establish a career in deep learning field. A million thanks to you.

Besides, I also must say thanks to my parents and my family for their love, support, and continuous encouragement throughout the course.

ABSTRACT

Generally, skin cancers, especially melanomas, have placed a huge health burden throughout the world, with the occurrence of more than 123,000 new cases annually. Early detection of melanoma is critical in preventing the progression of this disease into its invasive stages. However, most people are not giving enough attention to minor skin changes. Sometimes it is even difficult for doctors to distinguish between benign and malignant skin lesions. This study tries to solve it by proposing an automatic deep learning system for segmentation and classification in skin lesion images. This paper proposes a system that incorporates the use of a U-Net-based CNN and DeepLabV3, which proves helpful in the segmentation of an image in such a way that accurate mask. Furthermore, with the helping of the classifier model (ResNet-18) to classify the moles condition such as benign or malignant. The implementation of the system will be foreseen to enhance the diagnostic process, minimizing the time and difficulty brought forth by current methods, including invasive procedures and waiting for test results. This system uses a huge dermatological image database in order to apply deep learning methodologies for classifying a skin lesion with high precision. It will also be able to separate melanomas into either benign or malignant. The proposed automated system can lead to early diagnosis of the disease, which helps in effective early treatment, hence reducing the skin cancer burden. Moreover, the developed system will allow for patients and physicians to upload images through user-friendly web interface showing immediate real-time analysis and diagnosis. This is intended to run smoothly on the web interface, making it accessible both in clinical settings and possibly integrated into web interface for remote diagnostics. The research is done to show how deep learning can radically enhance the speed and accuracy of skin cancer diagnosis through segmentation and classification of skin lesions. If this can be affected with not much loss of time, it could save many lives by early detection and intervention.

Area of Study: Deep Learning Segmentation, Classifier Model

Keywords: Automated System, DeepLabV3, Custom U-Net, ResNet-18, Real-time analysis

TABLE OF CONTENTS

LIST OF FIGURES

LIST OF TABLES

LIST OF SYMBOLS

x	X variable
e	Euler's number

LIST OF ABBREVIATIONS

5G	Fifth Generation
UV	Ultraviolet
CNN	Convolutional Neural Network
U-Net	Convolutional Networks for Biomedical Image Segmentation
CAD	Computer-aided Systems
AI	Artificial Intelligent
PRISMA	Preferred Reporting Items for Systematic Reviews and Meta-Analysis
RoB	Risk of Bias
IoMT	Internet of Medical Things
ASPP	Atrous Spatial Pyramid Pooling
IoU	Intersection over Union
SSIM	Structural Similarity Index
DDN	Deep Neural Network
CRF	Conditional Random Fields
V3	Version 3
ROI	Region of Interest

Chapter 1

Introduction

In this chapter, we present the background and motivation of our research, our contributions to the field, and the outline of the thesis. Nowadays, most of the people was start to pay attention on their health and body condition when their body condition is getting worse but most of them were not paying attention on their skin condition. The major health problem was included skin cancer with over 123,000 newly diagnosed cases worldwide in every year [1]. Melanoma is the deadliest from the skin cancer, it had caused 9000 death in the United States each of the years [1]. Additionally, Melanoma only had 1% of skin cancer but it caused most of death from the skin cancer [2]. In United State in 2022, there had almost 100,000 new case was diagnosed that they had invasive mole, and caused 7650 death from the melanoma [2]. Furthermore, the melanoma can be removed before the skin lesions and become skin cancer. The main factor that caused of the skin cancer is UV exposure [3]. Unfortunately, although most of the people can observe the existence of mole but they are not able to differentiate that the mole is benign or malignant. Melanoma is one of the types of the skin lesion that mainly focused by most of the people now.

Therefore, an automated system for segmenting skin lesions using deep learning is very important by assisting people to differentiate the stage of the skin lesion and easier for the doctor to give an advice for the patient to do a surgery to remove the mole or not.

Furthermore, this project is aim to develop an automated system for segmenting and classify the skin lesions using deep learning. Deep learning is a machine learning that using the algorithms and function of human brain, especially neural networks. It designed to learn from the big data to make a decision by analyzing big data to ensure the decision that make by the machine is the greatest one. Not only that, the features like image, sound text and so on also able to automatically extract by deep learning machine to perform the task like detection, generation and classification. Furthermore, deep learning is important in image segmentation because it able the classify the big data. For example, by analyzing the image of the skin lesion in the database, it able to classify the type of the skin lesion by comparing the image that inside the database. Nevertheless, by using deep learning in image segmentation, it able to increase the accuracy of the diagnosis. For instance, it will compare with a big data and analyze it, in this process it will compare a big amount of data and able to low down the mistake when classify the type of the kin lesion. The following factor that deep learning is important in image segmentation was it can handle complex data. For example, the image of the skin lesion might just have minor changes such as melanoma. In the process on differentiate the mole is benign or malignant, the mole might just change it shape, color and so on. All these changes were minor and difficult to identify by human. Due to this incident, deep learning can differentiate the mole is malignant or not by handling the complex data like the image of the skin lesion with a minor change to give a more efficiency and accuracy decision for the patient or doctor. By develop this automated system it will be able to reduce the skin cancer cases by using the automated system in the early diagnosis of the mole and distinguish it was invasive or not.

1.1 Problem Statement and Motivation

In this day and age, most of the people are busy on earning money to create a comfortable life for their future. Majority of them does not pay attention on their skin condition. The long time expose under the sun will cause the existence of melanoma due to the UV of the sun was harmful to the human body. Skin lesion need to be diagnosed in the early stage of the period before it become skin cancer. The major factor is it only will be having some small exchange in the early stage for example, the asymmetry of the mole such as the irregular shape of it, the border of the melanoma such as the irregular border, color such as the multiple color of the melanoma, diameter of the melanoma such as more than 6 mm in diameter and evolving to be a melanoma such as change in appearance or symptoms [2]. Due to all of the changes all minor, most of the people are not able to observe the changes all the time and not possible to meet doctor every day. In this incident, the automated system can help the people to observe the change of the skin lesion by insert the picture of the skin, the system can help on differentiate the skin lesion is having potential to become skin cancer or not. Furthermore, most of the people are not able to differentiate the type of the skin lesion as well, although the doctor also will be needed to cut off a little piece of the skin from the mole to test the type of the skin lesion. This is a time and man power consumed process because it will take time to waiting the test report to differentiate the type of the skin lesion such as benign or malignant. The process on differentiates the type of the skin cancer also was difficult due to there had variety type of skin cancer for example Actinic Keratosis, Atyptical Moles, Melanoma, Basal Cell Carcinoma, Markel Cell Carcinoma, Squamous Cell Carcinoma and so on [3]. All these skin cancers will have different type of characteristics on skin. In this incident, by using the automated system, the system will able to differentiate the characteristics of the skin lesions and define it is benign or malignant in a short period of time and helping the doctor to give a suitable advice for the patient after getting know the type of the skin lesion.

In motivation part, the aim of the thesis is to propose an automated system for segmenting skin lesions using deep learning and the goal is to accurately identify and lesion in dermatological images to aid the early diagnosis and treatment of skin conditions, including skin cancer. This automated system will also be using the dataset of the dermatological images to differentiate and identify the condition of the skin. A person that without a knowledge on automated system also able to use this system to observe their skin condition.

1.2 Objectives

The aim of the thesis is to develop an automated system for segmenting skin lesions using deep learning. The first objective is to **developing an automated system for segmenting skin lesion using deep learning**. For instance, the pretrained model is not implemented, this project will be developing an automated system by implemented the DeepLabV3 model and UNET model into it. Furthermore, after implementation the DeepLabV3 and UNET model, the segmentation process will be handling by these two models. Due to the existences of two models, the accuracy will be having conflict. Therefore, the automated system will automatically compare both of these models and find out the most accuracy one. Through this process, it can ensure there will be having two models prepared to do the segmentation task and always provide the model that having high accuracy between these two models to the user. The second objectives for this project are to **implementing a classifier model into the automated system to classify the moles**. For example, the procedure on identifies the skin are benign or malignant is compare the image that had included in the database and based on the experience of the machine on learning form the diagnosis frequency. After the segmentation part complete, it will process to a classifier model to identify the skin condition like it is benign or malignant. The automated system will be able to complete the diagnosis in less time thanks to this procedure. Furthermore, the automated system can assist the physician or patient in precisely identifying skin diseases to facilitate an early diagnosis and determine a suitable course of therapy by using the dermatology image stored in the database. Due to this incident, the automated system only requires the user to provide the image of the skin for diagnosis and this process just consumed a short period of time, then it will be able make the diagnosis process more efficiency and accurately. The third objective is to **create a user-friendly web interface to show the analyses classification result of the skin lesions**. For instance, the web interface will be allowed medical professionals and patient to upload the images for diagnosis and received the results directly. By implement this user-friendly web interface, it able to allow the patient use a clear and easy way to observe their skin and also helping the doctor to make an accurate decision when doing the early diagnosis and informed the patient whether surgery is necessary.

1.3 Project Scope and Direction

The aim of this project is to develop an automated system by implementing UNET and DeepLabV3 model to do segmentation on the skin lesion and do comparing between both of the model. For example, the automated system will be having two model inside it and the segmentation process will be handling by these two models. After these two models have finishing on the segmentation process the automated system will auto choosing the mask that segmentate with a higher accuracy to do the classifier process.

Secondly, this project also will be involving a classifier model. For instance, the classifier will be handling the classify process such as classify the mole is benign or malignant. This model able to use the image that classify by DeepLabV3 or UNET to do the classification process. This will ensure that the segmentation of the image able to use for classification purpose.

Furthermore, this project also will be having an optimistic user experience by creating a user-friendly web interface for them. For example, all these models will be able to use on a user-friendly website, the user only will be required to send their mole image, and wait for the result. All the process of the model will be done on backend.

Lastly, this automated system will be targeted the user that having a healthy concern on their skin condition and wish to get their skin condition result but not willing on spending money to do a medical check-up on their moles.

1.4 Contributions

The automated system for segmenting skin lesions using deep learning and monitoring the skin condition by insert the image inside the automated system. Both of the models will be able to segmentate the skin lesion with a more accurately ways. Secondly, the classifier model will be able to help the user to classify the skin condition. For instance, classifier model will be used to classify the moles is benign or malignant. Furthermore, is the user-friendly web interface will be able to give the user an easier way to use the automated system. All these criteria will be able to reduce the time on waiting the test report for the skin lesion because for the automated system the user will only require to insert their image and wait for the automated system to differentiate the image by comparing the dermatological image that had store in the database to identify this skin lesion had potential to become skin cancer or not. it can reduce the time on waiting the test report for the skin lesion because for the automated system the patient will only require to insert their image and wait for the automated system to differentiate the image by comparing the dermatological image that had store in the database to identify this skin lesion had potential to become skin cancer or not.

1.5 Report Organization

This report is organized into 7 chapters: Chapter 1 Introduction, Chapter 2 Literature Review, Chapter 3 System Methodology or Approach, Chapter 4 System Design, Chapter 5 System Implementation, Chapter 6 System Evaluation and Discussion and Chapter 7 Conclusion and Recommendation. The first chapter is the introduction of this project which includes problem statement, project background and motivation, project scope, project objectives, project contribution and report organization. The second chapter is the literature review to evaluate the strength and weakness of the existing segmentation and classifier model. The third chapter is the method and approach, system architecture diagram, use case diagram, activity diagram and project timeline. The fourth chapter had outline system block diagram, system components and specifications, model selection and architecture, data preprocessing, model training and tuning, performance evaluation of the model and flask and web interface deployment. The fifth chapter consists of hardware setup, software setup, setting and configuration, system operation (with screenshot) and implementation issues and challenges. The sixth chapter had included system testing, project challenges and objectives evaluation. The final chapter which is chapter 7 have consisted conclusion and recommendation.

Chapter 2
Literature Review

2.1 Previous works on Deep Learning

2.1.1 Machine Learning and Deep Learning Methods for Skin Lesion Classification and Diagnosis [3]

This literature is about the overview of how CAD and AI methods have been applied in dermatology to aid in the diagnosis of skin lesions, especially melanoma and other skin cancers [3]. It essentially focuses on the conventional approaches of machine learning and deep learning in the diagnosis of skin lesions. With a critical evaluation of the literature, it underlines the fact that the recent accomplishments of deep learning enhance diagnostic performance considerably. The traditional approaches are very much based on hand-crafted feature extraction, whereas deep learning automatically extracts features, and hence it leads to more robust and scalable solutions. Deep learning models, especially CNNs, have shown outstanding performance in skin cancer diagnosis, often at a dermatologist level of accuracy. The challenges are high on the other side: large and diverse data sets, handling racial biases, and generalization across different populations.

Other than that, is the methods that used by the author is the systematic review and they also had look at the original papers written English in the ScienceDirect, SpringerLink, and IEEE database [3]. The author had shown the PRISMA diagram to propose the transparency by reviews process, to ensure the reader understand how select the paper in the final step.

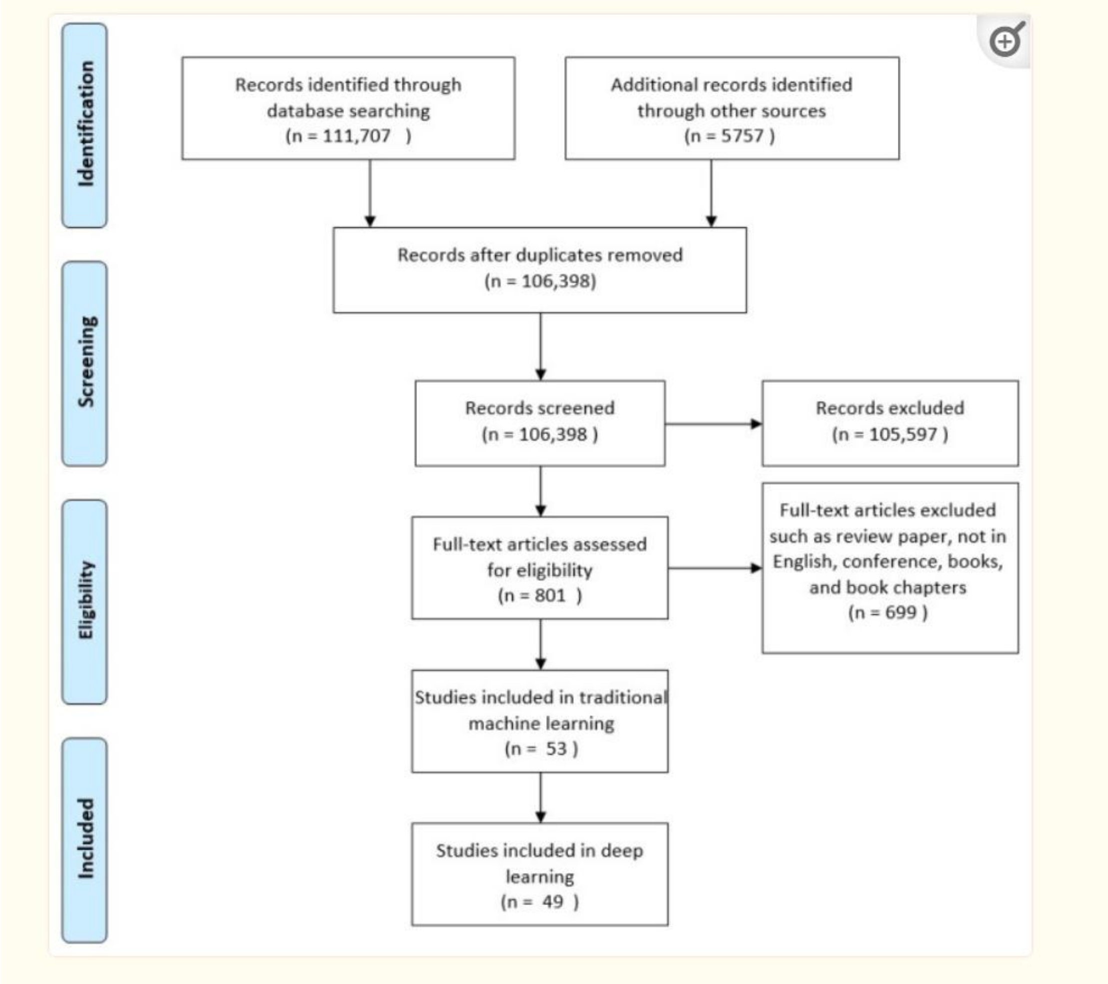

Figure 2.1 PRISMA diagram [3]

Based on the Figure 2.1 shown, in the PRISMA diagram it had shown the selection procedure. In the initial search, it had found out 111,701 of the literature sources which is suitable on the requirement of the search. There are 5757 records identified using other methods such as backward and forward snowballing were supplemented on these sources [3]. In the removal of the duplicate process, it will remove the duplicate record. After remove, it will be remaining with 106,398 records [3]. After that, is will identified 801 full-text articles. Lastly, 49 articles that using the deep learning were selected and 53 articles using traditional methods were selected [3].

2.1.2 Segmentation and Classification of Skin Lesions Using Hybrid Deep Learning [4]

This literature is about the improve the segmentation and classification of the skin lesions in the context of the IoMT by using hybrid deep learning approaches. Therefore, proposed hybridization of MRCNN for segmentation with ResNet50 for classification raises the overall performance and accuracy of skin lesion analysis manifold times [4]. The model performance is really remarkable regarding diagnosis of skin cancer in general and melanoma in an IoMT context, thus providing very enhanced diagnostic support to dermatologists.

Citation	Method	Dataset	Accuracy
[50], 2020	CNN	ISIC-2017	93.80%
[34], 2020	FCN	ISIC-2017	94.58%
[35], 2021	BAT	ISIC-2018	91.20%
[36], 2021	CNN	ISIC-2020	94.32%
[37], 2022	MS-RED	ISIC-2017	94.10%
[38], 2022	NCR-NET	ISIC-2017	94.01%
[39], 2023	MSFNet	ISIC-2018	92.17%
[40], 2023	CNN	ISIC-2017	91%
Proposed	**MRCNN**	**ISIC-2020**	**95.49%**

Figure 2.2: Testing result accuracy on MRCNN [4]

In the segmentation of skin lesions, the proposed hybrid model performed very well. For example, the Figure 2.2 shown that, the model reaching an accuracy as high as 95.49% on the MRCNN, and demonstrated increased performance compared to the existing state-of-the-art methods [4].

Citation	Method	Dataset	Accuracy
[18], 2020	MB-CNN	ISIC-2017	93.8%
[36]. 2021	Deep CNN	ISIC-2017	90.67%
[41], 2022	CNN	ISIC-2017	83.20%
[42], 2023	DSNN	ISIC-2019	89.57%
[43], 2023	YOLOv5	Self	79.20%
Proposed	**ResNet50**	**ISIC-2020**	**96.75%**

Figure 2.3: Testing results accuracy on RestNet50 [4]

Based on the Figure 2.3, the proposed model has achieved an accuracy of 96.75% on the ISIC 2020 dataset for classification on ResNet50, showing a vast improvement in comparison to traditional techniques [5]. Relatively, the results of segmentation and classification obtained with the proposed model were better in comparison to other techniques and thus may have clinical value for the diagnosis of skin lesions.

2.1.3 Skin Lesion Segmentation in Clinical Images Using Deep Learning [5]

This literature is about accurately segmenting skin lesions in clinical images of a method by using a deep learning approach. Furthermore, this literature had proposed highly effective deep learning method for skin lesion segmentation in clinical images such as preprocessing, CNN architecture and so on.

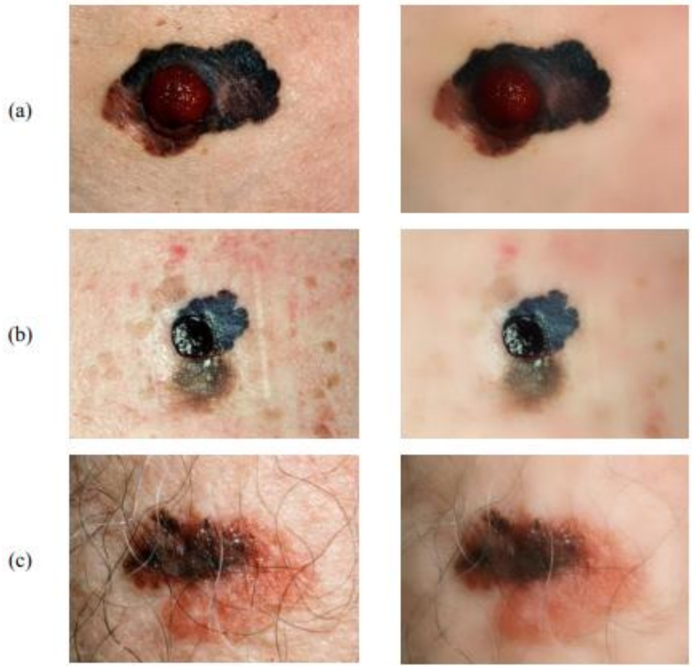

Figure 2.4: Left column are input images. Right column is preprocessed images, resulted from applying guided filter [5]

Based on the Figure 2.4 shown, it is the input images before and after preprocessed. For instance, these input skin images contain lots of artifacts, such as hair, light reflections, and uneven illumination; all these factors poison the segmentation process. To this end, a Guided Filter has been applied to the pre-processing part [5]. The Guided Filter serves as an edge-preserving smoothing operator that suppresses noise while preserving the border of the lesion. The guide for such filtering is the input image itself. This is demonstrated in figure 2.2.4 using dataset images of size 1640 × 1043 pixels [5]. This kind of filter smooths out the noisy textures which could mislead the segmentation process while maintaining the boundaries of the lesion. The output from this step, now pre-processed images, becomes the input to the CNN for further segmentation.

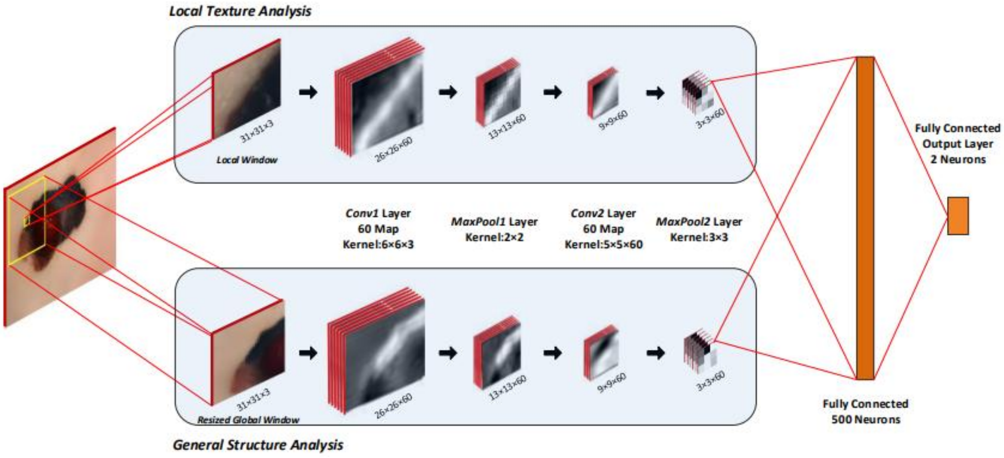

Figure 2.5: Architecture of the proposed CNN [5]

Based on the Figure 2.5 shown, CNN takes both the local and global patches around every pixel as its input. Images are first resized to 600 × 400 [5]. From every pixel, there are two patches extracted: a 31 × 31 local patch and a 201 × 201 global patches down sampled to 31 × 31 [5]. For the case where the part of the patch falls outside the image, reflection padding in multi-directions is performed. The CNN architecture fed with these patches in parallel consists of two convolutional layers, each using kernel sizes 6 × 6 × 3 and 5 × 5 × 60 [5], respectively, followed by max-pooling layers with kernel sizes of 2 × 2 and 3 × 3 [5], respectively. This allows the CNN to detect features across the image using 60 feature maps per convolution layer. Both local and global patch analysis results are combined via fully connected layers to provide the final label for the central pixel. The pooling layers reduce the number of learnable variables, enhancing learning efficiency by discarding positional information of the features.

2.1.4 Deep Learning Based Segmentation and Recognition of Dermatological Images [6]

The literature on deep learning-based segmentation and recognition of dermatological images is reviewed here. This review outlines how the techniques of deep learning apply to the field of skin disease image classification and its limitation. Furthermore, this paper discusses how AI-driven techniques help to improve accuracy and efficiency in analyzing skin disease images with lesser subjective biases in diagnosis. It has classified two types of methods in this paper: segmentation and classification methods. An example included in the segmentation method is DeepLabV3+ [6]. DeepLabV3+ is an advanced model with the ASPP module that can elevate the accuracy of semantic segmentation by acquiring the fine-grained details from an image, however, will call for higher computational resources in processing the images [6]. Another example, following in the classification methods, is MobileNet [6]. MobileNets are a family of lightweight deep learning models focused on efficient computation for mobile and embedded vision applications [6]. That is the reason for using depth wise separable convolutions to reduce computational costs, hence being suitable for skin disease classification.

2.1.5 Enhanced Skin Lesion Segmentation: DeepLabV3 and U-Net with Spatial Attention Mechanisms [7]

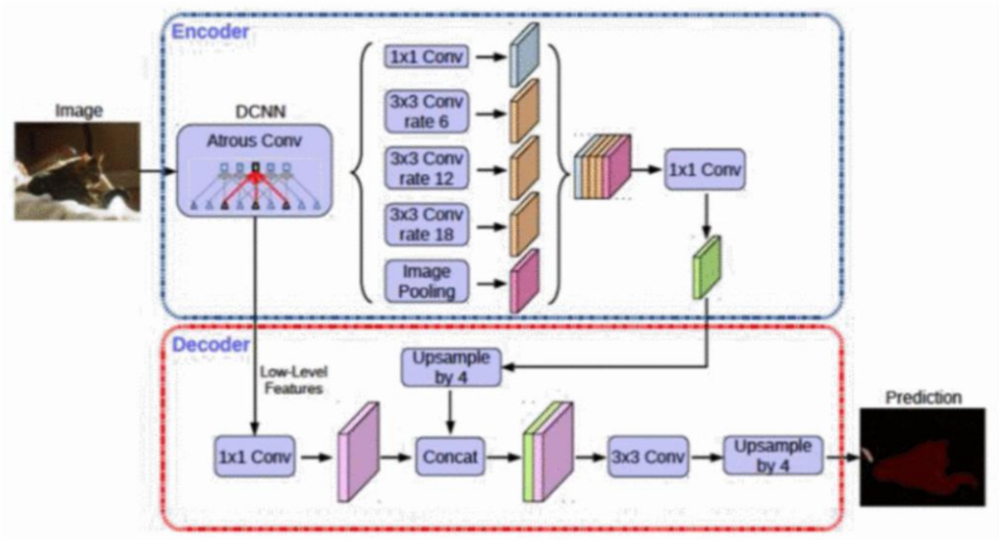

Figure 2.6: DeepLabV3 Diagram [7]

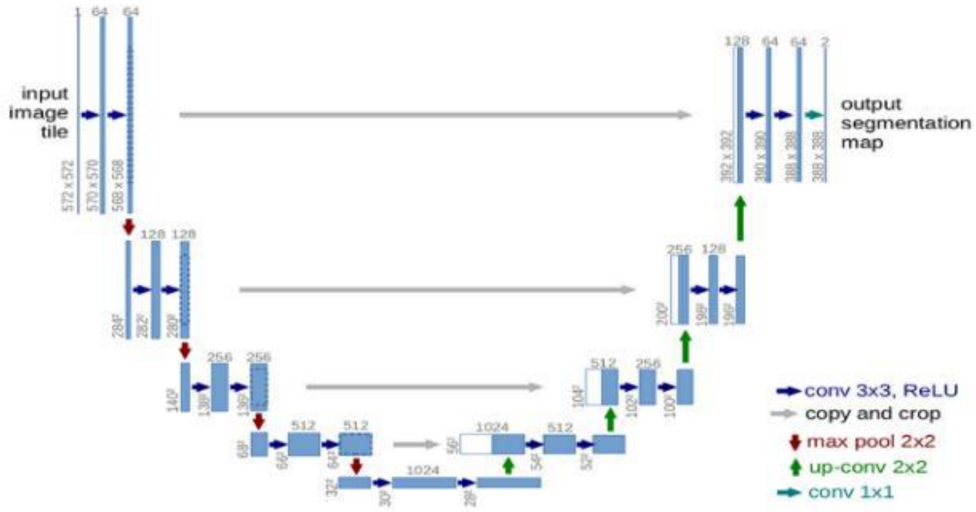

Figure 2.7: U-Net Diagram [7]

This literature is focused on the new enhancement ways for two main segmentation networks in skin lesion segmentation, which are DeepLabV3, as shown in Figure 2.6, and U-Net, as shown in Figure 2.7. DeepLabV3 and U-Net. DeepLabV3 is a Deep Neural Network (DDN) architecture; it usually uses a backbone that is similar to MobileNetV2 or Xception to capture the base characteristics, then applies atrous convolution to keep the characteristic images having a high resolution and extend the reception fields at the same time [7]. At its core lies the Atrous Spatial Pyramid Pooling (ASPP) module, which fuses the multiple atrous convolution branches, each with a different dilation rate but together with an image-level pooling branch to capture both fine details and global context. This will ensure that this network will not need to process afterwards and will be able to produce a smooth and accurate segmentation output end-to-end to efficiently enhance the segmentation accuracy in the complex background [7]. For U-Net, it was a convolutional neural network architecture that was created to segment the biological image in 2015 [7]. It applies a symmetric encoder and decoder (U-Net) design, which the contracting path through repeated 3x3 convolution and ReLU activations followed by max pooling to collect increasingly abstract context for the expanding path through using up sampling such as transposed convolutions and convolutions to recover the spatial resolution step by step. The decoder and encoder have a skip connection between their layers to fuse the high-resolution detail characteristics with deeper representations. This has kept the minor structure detail and mitigated the vanishing of the gradients. A final through 1x1 convolutions with the activation of Sigmoid produces pixel-wise probability maps to achieve the accuracy of the lesion's boundaries [7]. By adding lightweight spatial attention modules that allow the networks to dynamically concentrate on relevant lesion regions and inhibit background noise. The networks are trained and evaluated on the diverse ISIC thermoscopic dataset with aggressive augmentation policies, including rotation, scaling and color jitter. Both attention models utilize a hybrid loss combining Jaccard IoU (Intersection over Union) loss, SSIM (Structural Similarity Index) loss and focal loss for jointly optimizing pixel-level accuracy, structural fidelity, and focus on hard examples.

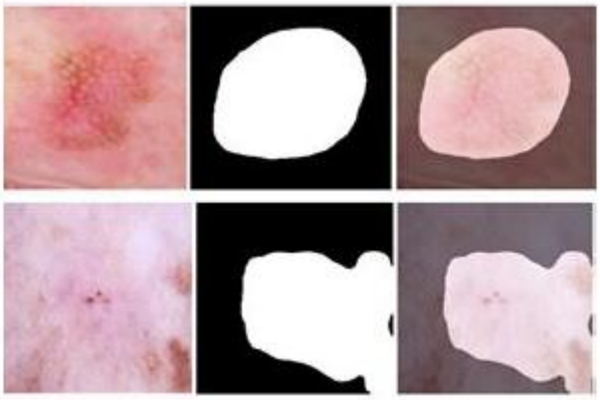

Figure 2.8: Segmentation with DeepLabV3 and Spatial Attention Mechanism (a) Input Image (b) Predicted masks (c) Overlay masks [7]

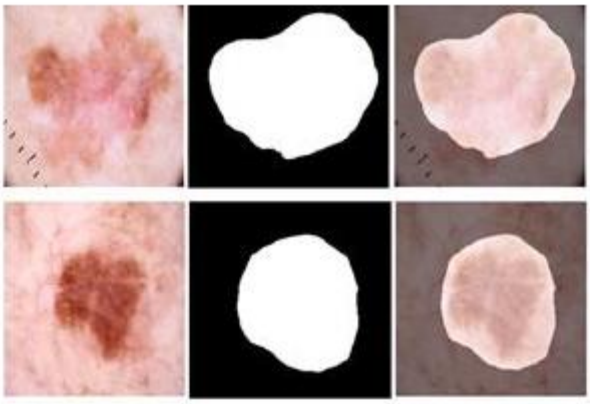

Figure 2.9: Segmentation with U-Net and Spatial Attention Mechanism (a) Input Image (b) Predicted masks (c) Overlay masks [7]

TABLE I. PERFORMANCE RESULTS WITHOUT ATTENTION MECHANISM

	IoU	Accuracy
DeepLabV3	0.76	0.80
U-Net	0.59	0.78

TABLE II. PERFORMANCE RESULTS WITH ATTENTION MECHANISM

	IoU	Accuracy
DeepLabV3	0.79	0.83
U-Net	0.62	0.80

Figure 2.10: Table of Performance Results without Attention Mechanism and Performance Result with Attention Mechanism from Figure 2.8 and 2.9 [7]

Other than that, based on Figure 2.10, which is the experimental results, it had shown that the attention mechanism improves DeepLabV3 IoU from 0.76 to 0.79 and its accuracy from 0.80 to 0.83, and it improves U-Net IoU from 0.59 to 0.62 and its accuracy from 0.78 to 0.80, with an overall improvement in performance that makes these architectures stronger and sufficient for early skin cancer detection in clinical settings [7].

2.1.6 Deep Residual Learning for Image Recognition: A Survey [8]

ResNet-18 is the entry-level member of the residual network family, introduced by He et al. in 2015 to address the degradation problem that plagues very deep convolutional nets. It begins with a single 7×7 convolution (64 filters, stride 2) and a 3×3 max-pool, then passes through four "stages" of residual blocks. Each block contains two consecutive 3×3 convolutions and each followed by batch-normalization and a rectified linear-unit activation—and adds its input directly to the block's output via a shortcut (or, when the block changes feature-map size or depth, via a 1×1 projection) [8]. ResNet-18 uses 2 such blocks per stage, with 64 to 128 to 256 to 512 filters (and spatial down sampling performed by stride-2 convolutions at the start of stages 2 to 4), for a total of 16 convolutional weight layers, plus the initial conv and a final fully connected layer, hence "18" layers. After the last block, global average pooling reduces each feature map to a single value before the final linear classifier. With only 11.7 million parameters, ResNet-18 trains quickly, generalizes well even on modest datasets, and runs efficiently on lightweight hardware, making it a go-to backbone whenever you need the benefits of residual learning without heavy computational cost.

2.2 Strengths and Weakness of These Existing Models

Table 2.1: Strengths and weakness of these existing models

Models	Strength	Weakness
U-Net [7]	• High accuracy segmentate: keeping detail when skip connection at spatial resolution, having excellent performance at limited medical sample. • End-to-end training: simple architecture, able to enhance segmentation without manual capture characteristics.	• High cost on memory and compute: multi-scale fusion and full-resolution convolutions required large memory of GPU. • Sensitivity of Noise: artifacts like glare and hair will interfere the segmentate of boundaries and lower down the accuracy.
DeepLabV3 [6]	• Multi-scale context aggregation via ASPP branches accompany with different dilation rates with adding image-level pooling, global semantics and capturing both details. • Without requiring CRF or other post-processing but still able having an end-to-end smooth output.	• Coarse boundaries: without decoder module, the border of the object might be refinement. • High compute cost: increasing memory and inference time due to the multiple parallel atrous.

DeepLabV3+ [6]	• Add decoder based on the fundamental of V3, sharper the boundaries. • Keep the strength of multi scale context on V3.	• Slower than V3 and use more memory
Mask R-CNN [4]	• Finds the lesion and segmentate it accurately in one time.	• Complex structure, slow in running and difficult to deploy.
ResNet50 [4]	• High accuracy on classifies benign and malignant.	• Big models, running slow on small devices or mobile phone.
MobileNet [4]	• Simple and fast, suitable for mobile terminal.	• Low accuracy compares to big model.
+Attention Mechanism [7]	• Enhance the performance: add 3% on accuracy and IoU by focus on legions of lesions.	• Will increase the diagnosis time and the cost of memory.
ResNet-18 [8]	• Small model and fast compute speed. • Easy to understand and deploy.	• Limited capacity of representational. • Sensitive to small dataset.

Chapter 3

System Methodology/Approach

The processes of the project were categorized into different phases in the development, which were project pre-development, data pre-processing, model training architecture building and data training, and prediction on test dataset. Due to the model is pre-trained, it will only be required to implement it into a useable model for the automated system. Other than that, this project also will develop a simple web interface for the user which able to let them interact with the system.

3.1 Proposed method/ Approach

The proposed method for developing an automated system for segmenting and classify skin lesions using deep learning will be based on the Agile methodology, and there will be six phases in the Agile methodology. The phrase will be followed by plan, design, develop, test, deploy and review. In the process of developing an automated system for segmenting skin lesions using deep learning, all the phrase in the Agile methodology is important to ensure all the progress can be controlled and accomplished on time.

Figure 3.1 Agile Methodology Phrase

Based on Figure 3.1, it shows the first phrase is 'plan'. In the plan phase, we will be required to identify the requirements for this project to create a functional automated system for the user that has a skin lesion problem to segment the skin lesion and doctors for reference, define the skin lesion type (benign or malignant), and share the result with the probability of being malignant for the user. This phrase will be having functional requirements and non-functional requirements. For the functional requirement, upload a clear skin lesion image (at least 720p resolution), perform a segmentation task that is done by DeepLabV3 and U-Net, evaluate the segmentation using the Dice score (if the ground truth is available), classify the type of skin lesion (benign or malignant) with ResNet-18, and lastly display the result. The result will include the segmentation overlay, classification, and confidence level (>=85%; high confidence for malignant, <=15%; high confidence level for benign). The non-functional requirements are to achieve a DICE score of >80% average, provide a user-friendly web interface and support real-time processing for uploaded images.

Other than that, in the second phase, which is the design phase, we will develop a system architecture such as frontend and backend and define components of the systems such as the pre-processing pipeline (the user uploads JPG/PNG, then it is resized to 256x256 for segmentation), segmentation models (DeepLabV3 and U-Net), the evaluation module (comparison of the surface area of the segmentation overlay based on the dice score of DeepLabV3 and U-Net), the classifier model (ResNet-18) and Flash (hosting the automated system). In this phrase also will be going to draft the system diagrams, such as the architecture diagram and use case diagram.

The following phrase will be development phrase. In this phrase, will be implement segmentation models such as U-Net (custom implementation) and DeepLabV3 (PyTorch torchvision), implement classification model such as ResNet-18, integrate models into the Flask backend. For example, this project also will be going to used the Microsoft Visual Studio code to load the checkpoint of the segmentation and classifier model that had pretrained before to a file name **model.py** and will be host by the flask and the flask will be named as **app.py**, these two files will be stored in the folder name **FYP2-Test** folder. Not only that, in this project also will be configure file storage for uploads and results.

The testing phrase will be going to test the pretrained models. For example, upload an image for the models and use the ISIC dataset with ground truth masks. After receiving the uploaded images, the system will start to evaluate segmentation results with the Dice coefficient. Both of the models, such as DeepLabV3 and U-Net, will be compared. The output which has the higher dice score image will be chosen and sent to the classifier model to test the accuracy for benign and malignant prediction.

The deployment phrase will be delivered to the completed system for the user. In this phrase, we will package trained models into a file named **/model** directory in the **FYP2-Test** folder. The **/model** file will be storing all the checkpoints and the weights of the pretrained models. Since this system is going to host locally, then in this project, we will deploy the Flask app locally and provide a user interface for the user.

The last phrase is a review phrase. In this phase, we will compare the final results against the project objectives, identify strengths, weaknesses and improved areas of the system, and document issues such as malignant bias, dataset limitation and deployment constraints.

All these phrases had able to integrates the segmentation models such as DeeplabV3 and U-Net, and the classifier model such as ResNet-18 together successfully. Not only that, all the model also able to use on the user interface smoothly.

3.2 System Architecture Diagram

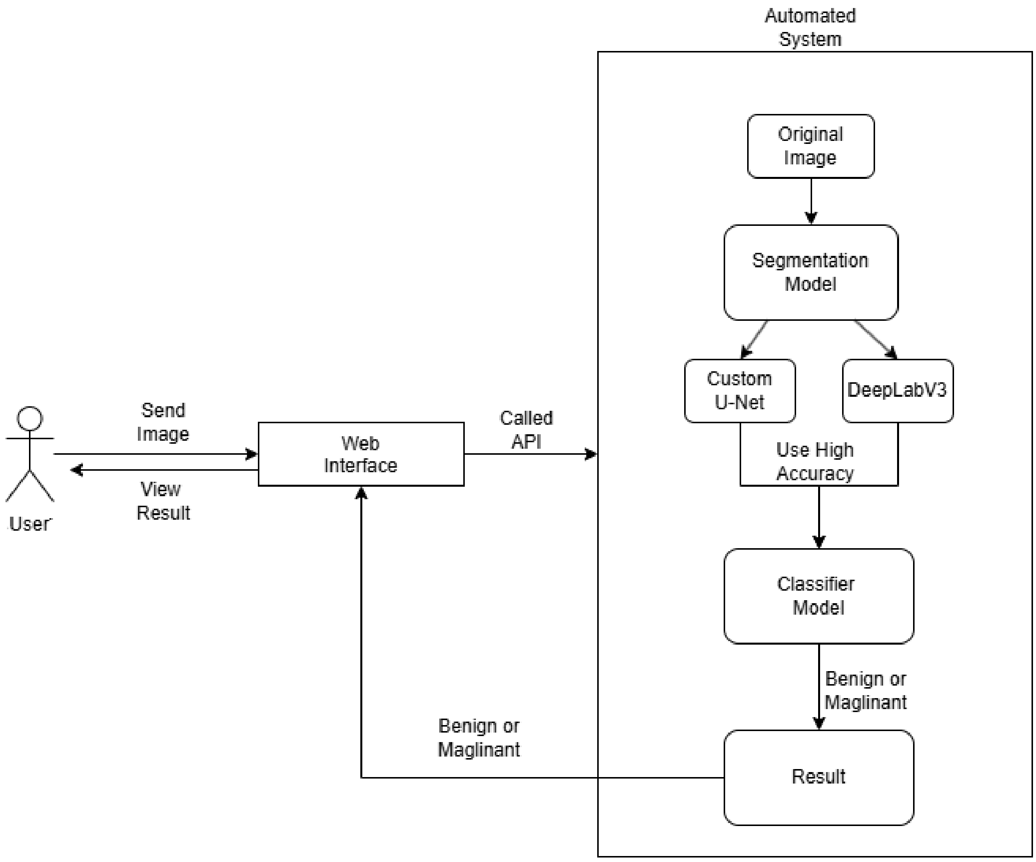

Figure 3.2: System Architecture Diagram

Based on the system architecture diagram showed at Figure 3.2, there will have several components will be exist in this project. First, it will be having a web interface as a frond end for the user to import their image inside the web. Besides, for the back end is the automated system which include segmentation model (Custom U-Net, DeepLabV3) and the classifier model (ResNet-18). The web interface will be built on the Microsoft Visual Studio Code and the main service which is segmentation and classification part also will be proceed on Microsoft Visual Studio Code too.

The purpose of the web interface is to let the user input their moles images on their skin. Then, the web interface will be able to fetch the image to the back end by calling the API of the back end to do the segmentation and classification task. Not only that, the result after classification also will be fetch back to the web interface to let the user to view the results.

Furthermore, is the element inside automated system. After receiving the input image, the automated system will load the images by read the RGB skin lesion photo (shape HxWx3) provided by the user. The following step is the normalize and resize for segmentation by resize to a size that able to match the segmentation network's expected input which is 256x256, scale the pixels value to (0,1), and additionally subtract it the same mean or divide by the same standard deviation used during training, then convert a shape 1x3x256x256 (batch x channels x height x width) of the tensor. Then, it will able to past to Custom U-Net and DeepLabV3 to do the segmentation part. In the segmentation part, both of the model will be segmentate in the same time and the accuracy will be list out, the mask that segmentate by the model that have a high accuracy will be choose to use in classification part. The higher dice score will be chosen because it has a major overlay on the original skin, so the comparison part the higher dice score image will be chosen. For the classification part, the classifier model will crop the patch into a second network (ResNet-18) that's been pretrained to classify on benign or malignant. It will give a score from 0 to 100 percent, if the probability is over 70%, it will be defined as malignant, otherwise will be benign. Due to the over fitting problem, the probability on defines the skin is malignant or benign is 70%. For example, more than 70% will be consider as malignant. Apart from that, there also have added a confidence level in the classifier model. For instance, the threshold of benign had been set to 0.15 and the threshold for malignant had been set to 0.85, which mean the confidence level on define the benign or malignant will be based on the threshold. There will be having three confidence level such as "**Low**", "**Medium**" and "**High**". When the classifier probability less than 0.05 will be showing "**High**" confidence level on diagnosis the skin is benign, if more than 0.05 but less than 0.15 will be "**Medium**". When the classifier probability more than 0.95, the confidence level on diagnosis it is malignant will be "**High**", otherwise the probability that between 0.85 to 0.95 will be state as "**Medium**". For the "**Low**" confidence level is to manage the uncertainty, for example when the classifier probability more than 0.60 will be "**Low**" for malignant, otherwise "**Low**" for benign.

Last but not least, after getting the result from the automated system. Then, the result will be fetching back to the web interface to let the user able to view the result.

Data Flow in this System Architecture Diagram:

Target users: User that having health concern on their moles

1. Web interface: User will be required to open the web interface.
2. Upload photo: User will need to upload a high-resolution photo on their moles.
3. Segmentation phrase: The upload photo will fetch to the automated system and start to do segmentation process. This process will be done by DeepLabV3 and Custom U-Net model.
4. Comparison phrase: The model that segmentate the photo with a high accuracy will be chosen.
5. Classification phrase: The chosen photo will be classified as benign or malignant.
6. Result shows: The final result will be show.
7. Fetch back results: The results will be fetching back to the web interface.
8. View result: User will be able to view the result through the web interface.

3.3 Use Case Diagram

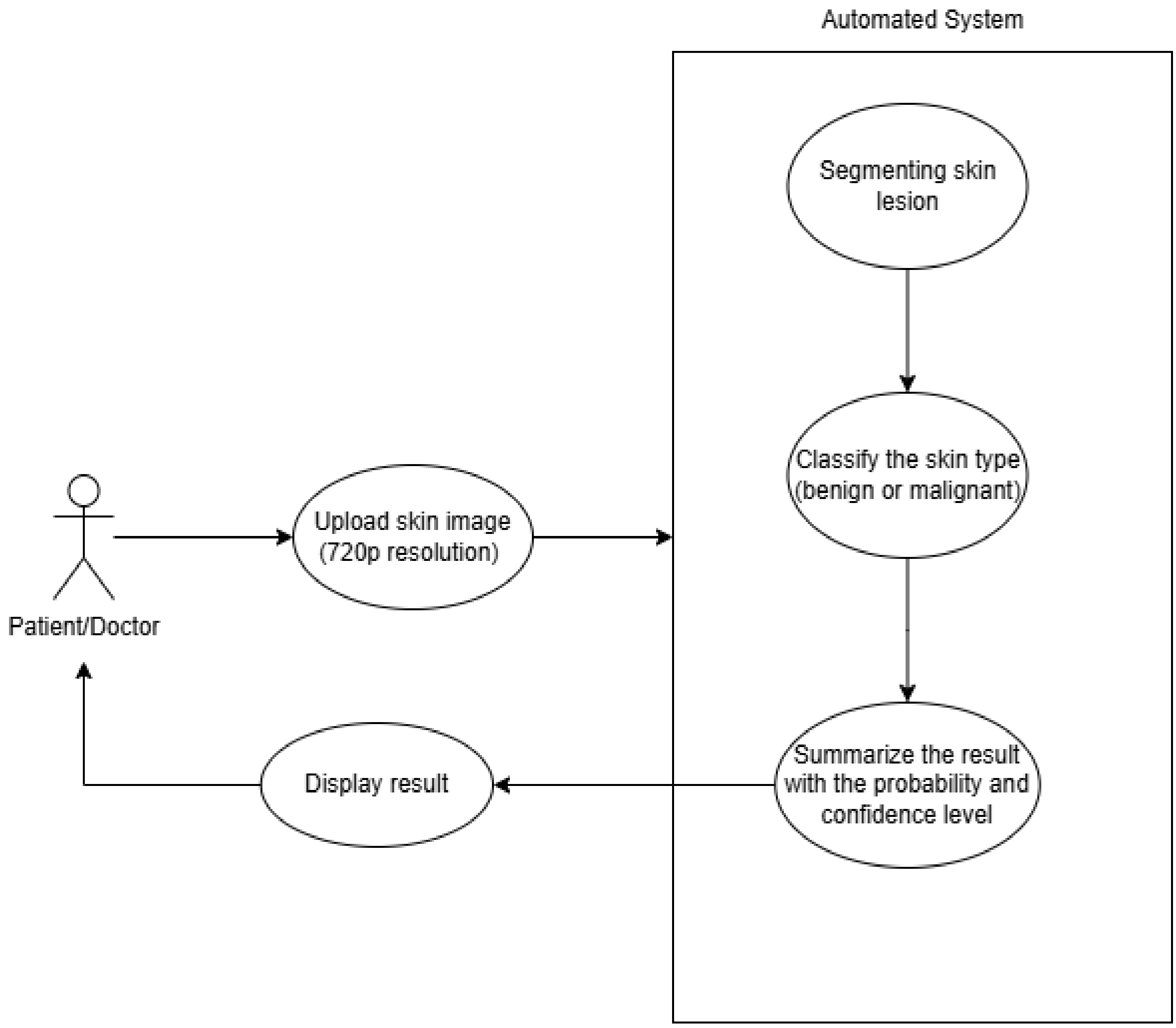

Figure 3.3 Use Case Diagram

Based on Figure 3.3, it had shown the use case diagram of the automated system for segmenting and classify skin lesion using deep learning and outlines the ways that patient or doctor interact with the automated system. First, patient or doctor will able to upload the skin lesion image. Then, the image will be sent to the automated system to segmenting the skin lesion image. The system will do comparison between two segmentation model which is DeepLabV3 and custom U-Net and choose the higher dice score image, then fetch to the classifier model. After the classify the skin lesion type such as benign or malignant, the result will be show accompany with the malignancy probability and confidence level for the user and doctor as a reference on the result through the web interface.

3.3.1 Use Case Description

Table 3.1 Use Case Description for “Upload Image” Use Case

Use Case ID	UC1	**Use Case Name**	Upload Lesion Image
Actor		Patient/Doctor	
Purpose		To allow user upload a skin lesion image for segmenting and classifying.	
Precondition		User have access to the webpage.	
Postcondition		Image is uploaded to server for processing.	

Table 3.2 Use Case Description for “Segmenting Image” Use Case

Use Case ID	UC2	**Use Case Name**	Perform Segmentation
Actor		Automated System (invoked by patient/doctor)	
Purpose		Generate lesion masks using DeepLabV3 and custom U-Net.	
Precondition		User have uploaded the image.	
Postcondition		Segmentation masks stored and ready for evaluation.	

Table 3.3 Use Case Description for “Evaluate Segmentation” Use Case

Use Case ID	UC3	**Use Case Name**	Evaluate Segmentation
Actor		Automated System (invoked by segmenting the image)	
Purpose		Choose the image that have higher dice score or larger overlay area between two segmentation model.	
Precondition		Image had been segmentate by the model.	
Postcondition		Higher dice score segmentation image had been chosen and ready for classification.	

Table 3.4 Use Case Description for "Classify Lesion" Use Case

Use Case ID	UC4	Use Case Name	Upload Lesion Image
Actor		Automated System (invoked by evaluation segmentation image which get the higher dice score)	
Purpose		Classify the skin lesion type such as benign or malignant.	
Precondition		The higher dice score segmentation image had been chosen.	
Postcondition		Red overlay image based on the original image, skin lesion type accompanies with the malignancy probability and confidence level are ready to display.	

Table 3.5 Use Case Description for "Display Result" Use Case

Use Case ID	UC5	Use Case Name	Upload Lesion Image
Actor		Automated System (invoked by classify lesion)	
Purpose		Display a result with a more detailed information.	
Precondition		The skin lesion type already classify clearly.	
Postcondition		A summary of a result will be display.	

3.4 Activity Diagram

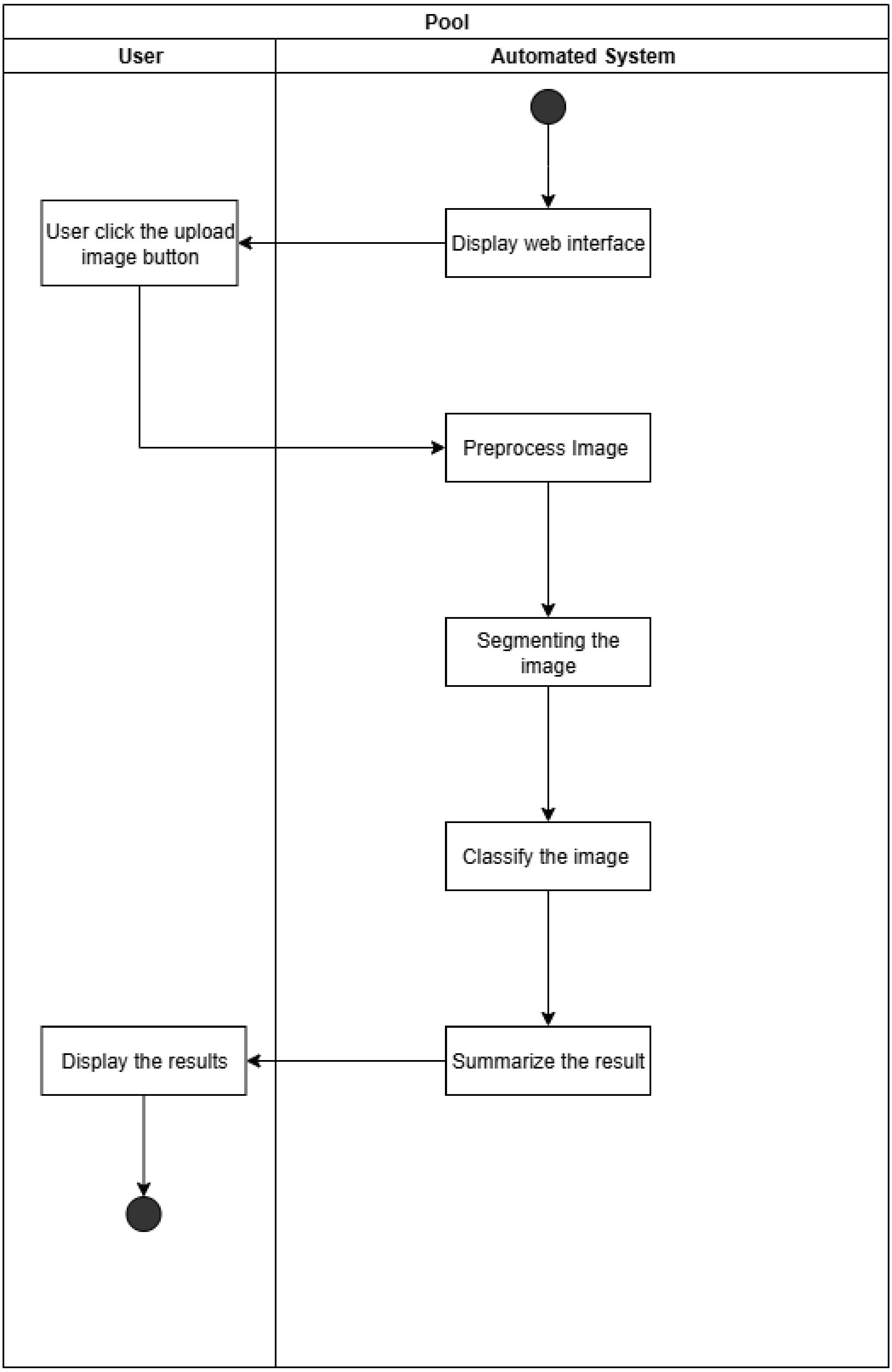

Figure 3.4 Activity Diagram for the Automated System

Based on the Table 3.6 shown, it was the activity diagram of the automated system which had provided details on how to interact with the automated system and how the system work. In the initial stage, when the user who is patient and doctor which represented by the dark circle clicks the ‘Upload Image” button in the web interface which locally host. The system will start to preprocess the image. In the preprocessing image part, all of the input image will be resized to 256x256 pixels since the models such as DeepLabV3 and custom U-Net only accept tensors of shape [1x3x256x256] and the mean = [0.485, 0.456, 0.406] and standard deviation = [0.229, 0.224, 0.225]. This setting is to ensure to compatibility with the pretrained backbones such as ResNet-18. After preprocess the images, the segmentation models (DeepLabV3 and custom U-Net) will start to segmentate the image and the image that had higher dice score or bigger red overlay will be chosen for accomplish classification task. The following stage is classifying the image, for instance based on the chosen image, the classifier model (ResNet-18) will be proceeding to classify the skin lesion type (benign or malignant). The following summarize the result and display the result for the user. The results will be included:

- Original image and red overlay image (To show the user the comparison of the image)
- Skin lesion malignancy probability.
- Skin lesion type (benign or malignant)
- Confidence level (Low, Medium, High)

Then, all these results information will be show to the user after the summarization of the results. All the information will be able to provide by the automated system for the user as a reference.

3.5 Project Timeline

Task Description	Week 1	Week 2	Week 3	Week 4	Week 5	Week 6	Week 7	Week 8	Week 9	Week 10	Week 11	Week 12	Week 13	Week 14
Revise on the FYP1 protocol														
Develop web interface														
Set up a flask														
Implement the flask into the web interface														
Writing FYP2 report														
Prepare for FYP2 presentation														

Figure 3.5 Final Year Project 2 Timeline

Chapter 4

System Design

4.1 System Block Diagram

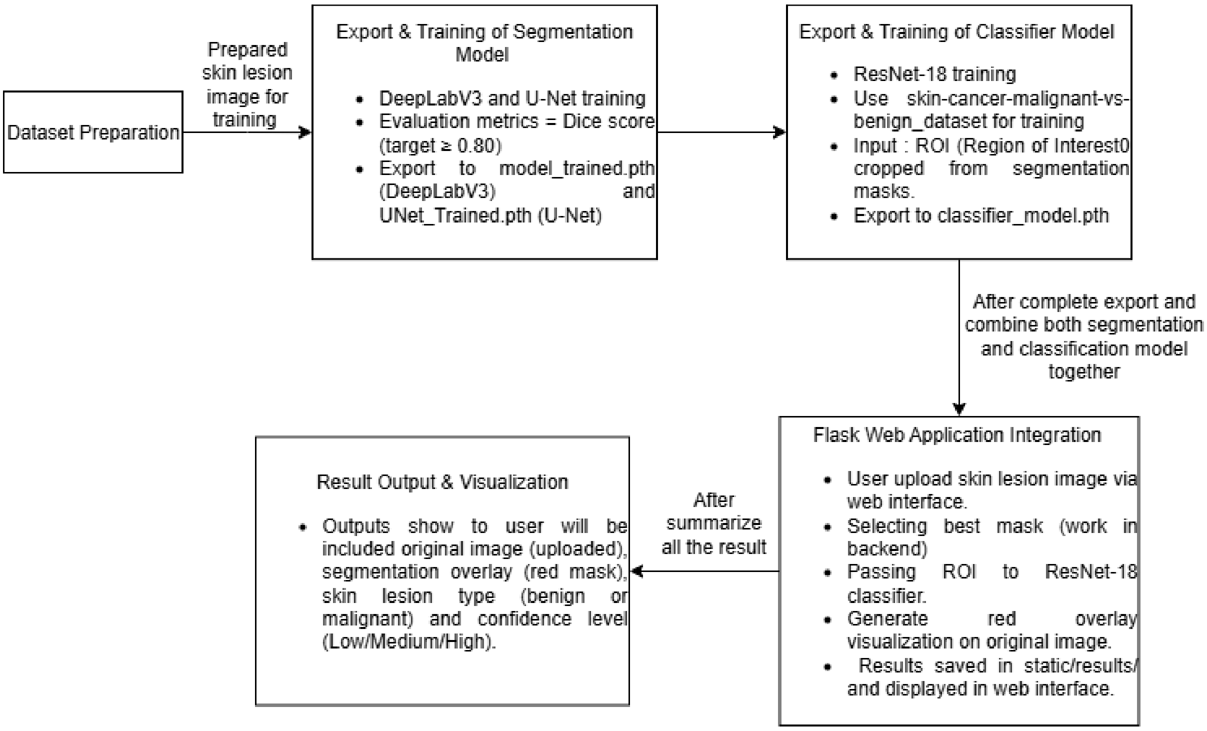

Figure 4.1 System Block Diagram

Based on Figure 4.1 shown, the system block diagram for the automated system for segmenting and classifying skin lesions using deep learning has included the procedure for the preparation of the dataset, training and export of the model, Flask web application integration and result output and visualisation. In the dataset preparation part, there are two sets of datasets that will be required to be prepared before proceeding to train the model. The first dataset that needs to be prepared for segmentation model training is the dataset used for segmentation model training (DeepLabV3 and custom U-Net), which is the ISIC 2018 Task 1 dataset. This dataset included 2594 of the skin lesion images, each paired with their corresponding ground truth segmentation mask that has been provided in the dataset. The ground truth is available in the dataset for training and is used to supervise the segmentation models. In addition, 100 validation images and 1000 test images are provided. The other dataset that needs to be prepared is “Skin Cancer: Malignant vs Benign”, which is available on Kaggle.com. This dataset is used for classifier model training. In this dataset, the images for the training set are 2637 images with a subfolder of benign and malignant and 1000 images for the testing set, which is also split into benign and malignant. Furthermore, there is the segmentation model training and export part. Since the segmentation model is a pretrained model, the task that needs to be accomplished in this project is to run again the model training part, observe the accuracy of the segmentation and store the checkpoint of the model. After completely running the pretrained model, this segmentation model had achieved a dice score that was greater than 0.80, which for the custom U-Net average is 0.833 and for DeepLabV3 is 0.847. This dice score has been done by 20 epochs in train mode and validation mode. Therefore, the checkpoint of the segmentation model (DeepLabV3 and custom U-Net) will be stored. The following part is classifier model part. The classifier model that used is ResNet-18. Since, the classifier model also is a pretrained model, therefore the task that need to done at this part is run again the code, observe the accuracy and export the checkpoint. After completed running the classifier model, it having 93.59% of the accuracy which done by 10 epochs. Not only than that, the following part is flask web application integration. This part will be combining the segmentation and classification model by load the checkpoint of both models. Additionally, set a flask that host locally with the interaction of web interface which able to provide a platform for the user (patient or doctor) to upload the skin lesion images for segmentate and classify. Last but not least, after summarize all the result, the result will be display on the web interface for the user as a reference with a detailed information. The detailed information has included original image (uploaded), segmentation overlay (red mask) for comparison, skin lesion type (benign or malignant) and the confidence level (Low, Medium, High).

4.2 System Components Specifications

4.2.1 Hardware

The hardware involved in this project is computer. A computer issued for the process of data pre-processing and model training. For instance, implement custom U-Net for features extraction for the model training part. For data pre-processing, it will involve image resizing, normalization, augmentation and so on. Furthermore, computer also will be used to process the classifier model on classify the skin lesions and creating the web interface.

Table 4.1 Specifications of laptop

Description	Specifications
Model	Asus TUF Gaming FX505DT_FX505DT
Processor	AMD RYZEN 5 3550H
Operating System	Windows 11
Graphic	NVIDIA GeForce GTX 1650
Memory	24GB RAM
Storage	512 GB SSD

4.2.2 Software

Various development tools and frameworks have been used for the automated skin lesion segmentation and classification system. Publicly available datasets like the ISIC 2018 Challenge dataset from Kaggle can be utilized; these will have all sets of dermatological images which shall be required during the training and validation of deep learning models. All the model will be train on the Jupyter Notebook. Other than that, the Microsoft Visual Studio Code also will be utilize to implementing these two models and the classifier model as well. Not only that, web interface also will be created in this software.

Table 4.2: Software Specification

Specification	Description
Jupyter Notebook	• Run the pre-trained segmentation model to save the checkpoint and weight.
Microsoft Visual Studio Code	• Implement the pre-trained model by load the checkpoint and weight. • Run the pre-trained classifier model to save the checkpoint and weight • Creating simple web-interface.
Storage	• Minimum 4GB of available disk space
Display Resolution	• Based on the image input by the user. Recommended resolution is 1280x720p
RAM	• Minimum 8GB

4.2.3 Elements of Software

Elements of software is the things that need to be prepared before deploy these models and flaks on the laptops.

4.2.3.1 Programming Language

- Environment: Python
- Version: 3.9.18
- Purpose: Selected as the development language because it was efficient on deep learning and web application. It also compatible with the latest PyTorch and Flask, this can ensure a seamless integration between the models and the web interface.

4.2.3.2 Libraries and Framework

PyTorch

- Role: Core of Deep Learning framework for training.
- Purpose: Load and run the segmentation models (DeepLabV3 and custom U-Net model), implement the classifier model (ResNet-18) for classify the skin lesion type (benign or malignant).

Torchvision

- Role: Provides pretrained models and utilizes the transformation of image.
- Purpose: ResNet-101 backbone on DeepLabV3 (segmentation purpose), ResNet-18 classifier model and the function for image preprocessing such as resizing, tensor conversion and normalization.

OpenCV

- Role: Image processing in a traditional way and the refinement of the mask.
- Purpose: Remove noise from predicted masks, extracting lesion bounding boxes and lesion contours for ROI (Region of Interest) classification, overlay a red mask on the original image for visualization

NumPy and PIL (Python Imaging Library)

- Role: Array manipulation and image input and output.
- Purpose: Coverts between PIL images and NumPy arrays, handling cropping, reshaping and normalization of pixels, save overlays result in a standard format.

Flask

- Role: Interaction system between the models and web interface
- Purpose: Provides the web interface for the user to upload image, fetch the image for the models to do segmentation and classification in backend, user able to view the result of the skin lesion with detailed information (original image, red overlay image, skin lesion type, probability of the skin lesion, and confidence level).

4.3 Model Selection and Architecture

4.3.1 Custom U-Net (Segmentation Model)

The model architecture uses Convolutional Neural Networks, which can extract key features in images through successive convolutions and pooling processes. In this project, UNET architecture is applied [7], which is one of the most popular models in medical image analysis because of the nature of its structure: encoder and decoder are connected even for segmenting images accurately to a level of pixels. The software tools and architectures being integrated will make up a formidable system toward the correct diagnosis of skin conditions, while providing critical support to early detection and treatment planning. The integration of Kaggle's dataset, CNN models [9], and UNET architecture guarantees that the system will be both effective and scalable in a range of clinical applications. This project will be based on the concept of U-Net and used for the Custom U-Net.

4.3.2 DeepLabV3 (Segmentation Model)

The following model is the DeepLabV3 model also will be included in this project for segmentation. DeepLabV3 begins by passing the input image through a deep convolutional network backbone which is a popular option being ResNet-101, that extracts dense, high-level feature maps. They are then passed through the Atrous Spatial Pyramid Pooling (ASPP) module, where several parallel convolutions with different dilation rates encode lesion and background context at different scales. The different feature outputs are concatenated, projected with a 1×1 convolution, and up sampled to a 256×256 resolution. A final sigmoid activation converts the output of every pixel to a "lesion vs. background" probability, which you can threshold such as at 0.5 to obtain a binary segmentation mask. Due to using two model which is Custom U-Net and DeepLabV3 for segmentation and compare both of it will be more reliable.

4.3.3 ResNet-18 (Classification Model)

$$p = \frac{1}{1 + e^{-x}}$$

Figure 4.2 Formula of Sigmoid [9]

For the classifier model also will use the prepared data set from Kaggle such as Skin Cancer Malignant vs Benign. These data set will be used for training a classifier model which is ResNet-18 that able to classify the segmentation skin that done by U-Net or DeepLabV3. ResNet-18 is an 18-layer residual network built from "residual blocks": a pair of 3×3 convolutions whose output is combined with the block input by an identity shortcut so that gradients flow freely throughout the network. The blocks are arranged in four groups, with occasional down-sampling being brought about by having stride=2 for the first convolution in a block. After all the blocks, global average pooling condenses spatial information into one feature vector and feeds it to a last fully-connected layer. To predict skin lesions, you crop out the segmented region of interest (ROI), resize to 224×224, and pass it through ResNet-18. The calculation of the malignant will based on the formula that shown on the figure 3.1, let assume the logit x is 0.8, after insert the calculation in the formula, it will have a 0.6899 which is about 69% of the probability will become a malignant but only have low confidence level, as the condition that set in the classifier model when the p is greater than 0.6 but less than 0.85 will be considered as malignant in low confidence level. This confidence level had been added because need to deal with the over fitting problem. This had been adjusted when loaded it out on the **model.py** file. For instance, the threshold of benign had been set to 0.15 and the threshold for malignant had been set to 0.85, which mean the confidence level on define the benign or malignant will be based on the threshold. There will be having three confidence level such as “**Low**”, “**Medium**” and “**High**”. When the classifier probability less than 0.05 will be showing “**High**” confidence level on diagnosis the skin is benign, if more than 0.05 but less than 0.15 will be “**Medium**”. When the classifier probability more than 0.95, the confidence level on diagnosis it is malignant will be “**High**”, otherwise the probability that between 0.85 to 0.95 will be state as “**Medium**”. For the “**Low**” confidence level is to manage the uncertainty, for example when the classifier probability more than 0.60 will be “**Low**” for malignant, otherwise “**Low**” for benign.

4.4 Data preprocessing

4.4.1 Data preprocessing for Segmentation Models (DeepLabV3 and Custom U-Net)

```
[79]: ############################ Adapted from Practical Lab 7 ############################
#Initially the value in the masks are either 0 or 255, we convert the value of 255 to 1 in order to do the binary image segmentation task

def preprocess_mask(mask):
    if not isinstance(mask, tv_tensors.Mask):
        mask = tv_tensors.Mask(mask)
    # Convert non-zero values to 1
    mask.data[mask.data != 0] = 1
    return mask
```

```
[38]: ############################ Reference: Practical 7 ############################

#to show one sample
def show_sample(img, mask=None, unnormalize=False):

    if isinstance(img, tv_tensors.Image):
        img = img.detach().cpu()
        if unnormalize:
            img = img*torch.tensor(STD).reshape(-1, 1, 1) + torch.tensor(MEAN).reshape(-1, 1, 1)
        img = v2.functional.to_pil_image(img)

    if mask is not None:
        if isinstance(mask, tv_tensors.Mask):
            mask = mask.detach().cpu()
            mask = v2.functional.to_pil_image(mask)

        fig, axs = plt.subplots(1, 2, figsize=(5, 3))
        axs[0].imshow(img)
        axs[0].axis('off')
        axs[0].set_title("Image")
        axs[1].imshow(mask, cmap="gray")
        axs[1].axis('off')
        axs[1].set_title("Mask")
    else:
        plt.figure(figsize=(5, 3))
        plt.imshow(img)
        plt.axis("off")

    plt.tight_layout()
    plt.show()
```

Figure 4.3 Data Preprocessing Code Screenshot for Segmentation Model (DeepLabV3 and Custom U-Net) 1

Based on the Figure 4.3 shown, all these codes are implemented to binarize the ground truth masks, by converting values of 255 to 1 while remain the background as 0 for as a preprocessing function. In addition, a utility of visualization (show_sample) was created to display skin lesion image with the segmentation masks, to ensure the data quality and the inspection of the preprocessing correctness.

```python
############################ Adapted from Lab 7 ############################

#data augementation for training images and mask

train_transform = v2.Compose([
        v2.ToImage(),
        v2.Resize(size=(256, 256), interpolation=InterpolationMode.BILINEAR, antialias=True),
        v2.RandomHorizontalFlip(p=0.5),
        v2.RandomVerticalFlip(p=0.5),
        v2.RandomRotation(degrees=30),
        v2.ColorJitter(brightness=.3, contrast = 0.3),
        v2.GaussianBlur(kernel_size=5, sigma=(0.1, 2.0)),
        v2.ToDtype(torch.float32, scale=True), # convert to float32 and scale to [0, 1]
        v2.Normalize(mean=MEAN, std=STD)
    ])
train_transform
```

```
Compose(
      ToImage()
      Resize(size=[256, 256], interpolation=InterpolationMode.BILINEAR, antialias=True)
      RandomHorizontalFlip(p=0.5)
      RandomVerticalFlip(p=0.5)
      RandomRotation(degrees=[-30.0, 30.0], interpolation=InterpolationMode.NEAREST, expand=False, fill=0)
      ColorJitter(brightness=(0.7, 1.3), contrast=(0.7, 1.3))
      GaussianBlur(kernel_size=(5, 5), sigma=[0.1, 2.0])
      ToDtype(scale=True)
      Normalize(mean=[0.485, 0.456, 0.406], std=[0.229, 0.224, 0.225], inplace=False)
)
```

```python
############################ Reference: Practical Lab 7 ############################

#use to transform image and mask
img  = Image.open(os.path.join(images_dir, images_filenames[id] + ".jpg"))
mask = Image.open(os.path.join(masks_dir, images_filenames[id] + "_segmentation.png"))
mask = preprocess_mask(mask)
show_sample(img, mask)
```

```python
############################ Reference: Practical Lab 7 ############################

#transforming both image and mask together ## this contain color jitter
transformed_img, transformed_mask = train_transform(img, mask)
show_sample(transformed_img, transformed_mask, unnormalize=True)
```

Figure 4.4 Data Preprocessing Code Screenshot for Segmentation Model (DeepLabV3 and Custom U-Net) 2

Based on Figure 4.4 shown, there have various training images and segmentation masks were processed to increase the accuracy and prevent overfitting. The size of the images was resized to 256x256 pixels because the segmentation model only accepts these sizes of images, then using ImageNet statistics to normalized to match with the pretrained models. Other than that, data augmentation also included in this project, the tasks that it accomplishes is rotations plus or minus 30 degrees, flips randomly, contrast and brightness adjustments, and Gaussian blur. These transformations able to help the model on learn to manage various of changes such as orientation, noise and lighting. Most important part is both of the images and the mask were transformed together to ensure the alignment of lesion legions are correct.

```
[44]: ############################ Reference: Practical Lab 7 ############################

#data augementation for validation images and mask

val_transform = v2.Compose([
    v2.ToImage(),
    v2.Resize(size=(256, 256), antialias=True),
    v2.ToDtype(torch.float32, scale=True),
    v2.Normalize(mean=MEAN, std=STD)
    ])
```

```
[80]: ############################ Reference: Practical Lab 7 ############################

#data augementation for validation images and mask

test_transform = v2.Compose([
    v2.ToImage(),
    v2.Resize(size=(256, 256), antialias=True),
    v2.ToDtype(torch.float32, scale=True),
    v2.Normalize(mean=MEAN, std=STD)
    ])
```

Figure 4.5 Data Preprocessing Code Screenshot for Segmentation Model (DeepLabV3 and Custom U-Net) 3

Based on Figure 4.5 shown, the validation and test dataset only have the basic pre-processing processes applied without an additional augmentation. Each of the skin lesion images and segmentation masks was converted to tensor format and resized to 256x256 pixels, and ImageNet statistics were used to normalise it (mean = [0.485, 0.456, 0.406], std = [0.229, 0.224, 0.225]). This was not similar to the training set; there are no rotations, random flips or colour changes used, because the aim of validation and testing is to measure the true performance of the models on the data that had not been seen before.

```python
# A class for storing dataset
# During training will return mask, during testing no mask only resolution
class Cancer(Dataset):
    def __init__(self, images_filenames, images_dir, masks_dir, transform=None, test_mode=False):

        self.images_filenames = images_filenames
        self.images_dir       = images_dir
        self.masks_dir        = masks_dir
        self.transform = transform
        self.test_mode = test_mode

    def __len__(self):
        return len(self.images_filenames)

    def __getitem__(self, idx):

        # get image at position idx
        image = Image.open(os.path.join(images_dir, self.images_filenames[idx]+ ".jpg"))
        image = tv_tensors.Image(image)


        #during training return mask, testing dont assume a mask(only get back resolution)
        # for test mode, return the image and its actual resolution
        if self.test_mode:
            resolution = image.shape[1:] #(C,H,W)
        #   if the transformation pipeline is passed by user
            if self.transform is not None:
                image = self.transform(image)

            mask = Image.open(os.path.join(self.masks_dir, self.images_filenames[idx] + "_segmentation.png"))
            mask = preprocess_mask(mask)
            if self.transform is not None:
                _, mask = self.transform(None, mask)  # Apply transformation to mask (assuming it doesn't need any transformation)
            return image, resolution, mask

        # for train mode, return the image and its mask
        mask = Image.open(os.path.join(masks_dir, self.images_filenames[idx] + "_segmentation.png"))
        mask = preprocess_mask(mask)
        # if the transformation pipeline is passed by user
        if self.transform is not None:
            image, mask = self.transform(image, mask)

        return image, mask
```

Figure 4.6 Data Preprocessing Code Screenshot for Segmentation Model (DeepLabV3 and Custom U-Net) 4

```python
[48]: ############################ Reference: Practical Lab 7 ############################

#pass trainset to training loader
trainset    = Cancer(train_images_filenames, images_dir, masks_dir, transform=train_transform, test_mode=False)
trainloader = DataLoader(trainset, batch_size = BATCH_SIZE, shuffle=True, num_workers=2)

x_batch, y_batch = next(iter(trainloader))
print(f'{x_batch.shape = }')
print(f'{y_batch.shape = }')
```

```
x_batch.shape = torch.Size([12, 3, 256, 256])
y_batch.shape = torch.Size([12, 1, 256, 256])
```

```python
[85]: ############################ Reference: Practical Lab 7 ############################

#pass trainset to validation loader
valset      = Cancer(val_images_filenames, images_dir, masks_dir, transform=val_transform, test_mode=False)
valloader   = DataLoader(valset, batch_size = BATCH_SIZE, shuffle=False, num_workers=2)

x_batch, y_batch = next(iter(valloader))
print(f'{x_batch.shape = }')
print(f'{y_batch.shape = }')
```

```
x_batch.shape = torch.Size([12, 3, 256, 256])
y_batch.shape = torch.Size([12, 1, 256, 256])
```

```python
[73]: ############################ Reference: Practical Lab 7 ############################

#pass trainset to testing loader
testset     = Cancer(test_images_filenames, images_dir, masks_dir, transform=test_transform, test_mode=True)
testloader   = DataLoader(testset, batch_size = BATCH_SIZE, shuffle=False, num_workers=2)
img, res, mask    = testset[np.random.randint(len(testset))]
show_sample(img, mask, unnormalize=True)
print('Resolution of image', res)
```

Figure 4.7 Data Preprocessing Code Screenshot for Segmentation Model (DeepLabV3 and Custom U-Net) 5

Based on the Figure 4.6 and 4.7 shown, is a custom dataset class had been designed to let the image and mask pairing during the training and validation, while also support test mode for only input images are available. By combine this class with PyTorch DataLoader, the system had efficiently handled shuffling, batching and preprocessing, to ensure the data was fed into the segmentation models in a scalable and consistent manner.

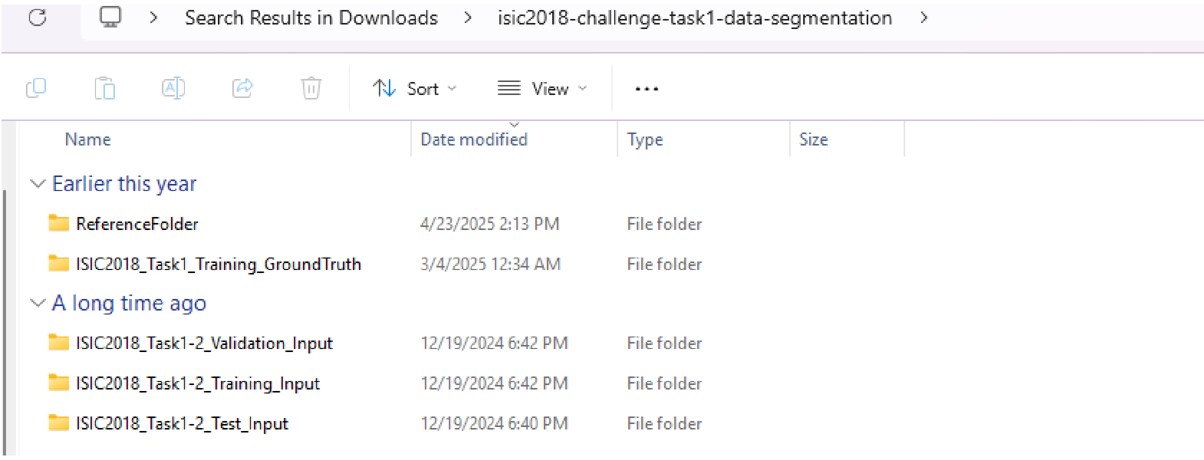

Figure 4.8 Dataset that used to trained segmentation models

Based on the Figure 4.8 shown, it had included all the dataset that used to trained the segmentation model in folder format. The folder had included the validation input, training input and the test input.

4.4.2 Data preprocessing for Classifier Model (ResNet-18)

```python
import os
from PIL import Image
import torch
from torch.utils.data import Dataset, DataLoader
import torchvision.transforms as transforms
import torch.nn as nn
import torch.optim as optim
import torchvision.models as models
import matplotlib.pyplot as plt

# ----------------------------
# 1. Custom Dataset Class
# ----------------------------
class SkinCancerDataset(Dataset):
    def __init__(self, benign_dir, malignant_dir, transform=None):
        """
        benign_dir: Path to benign image folder
        malignant_dir: Path to malignant image folder
        transform: Image preprocessing pipeline
        """
        self.transform = transform
        self.image_paths = []
        self.labels = []  # Benign: 0, Malignant: 1

        # Load benign images
        for filename in os.listdir(benign_dir):
            if filename.lower().endswith(('.png', '.jpg', '.jpeg')):
                self.image_paths.append(os.path.join(benign_dir, filename))
                self.labels.append(0)
        # Load malignant images
        for filename in os.listdir(malignant_dir):
            if filename.lower().endswith(('.png', '.jpg', '.jpeg')):
                self.image_paths.append(os.path.join(malignant_dir, filename))
                self.labels.append(1)

    def __len__(self):
        return len(self.image_paths)

    def __getitem__(self, idx):
        # Load image and convert to RGB mode
        img = Image.open(self.image_paths[idx]).convert("RGB")
        if self.transform:
            img = self.transform(img)
        label = self.labels[idx]
        return img, label
```

Figure 4.9 Data Preprocessing Code Screenshot for Classifier Model (ResNet-18) 1

```python
# ----------------------------
# 2. Define Preprocessing (consistent with training)
# ----------------------------
transform = transforms.Compose([
    transforms.Resize((224, 224)),  # Resize image
    transforms.ToTensor(),
    transforms.Normalize(mean=[0.485, 0.456, 0.406],
                         std=[0.229, 0.224, 0.225])
])

# ----------------------------
# 3. Create Dataset and DataLoader
# ----------------------------
benign_dir = r"C:\Users\win10\Downloads\skin-cancer-malignant-vs-benign_dataset\skin-cancer-malignant-vs-benign\data\train\benign"
malignant_dir = r"C:\Users\win10\Downloads\skin-cancer-malignant-vs-benign_dataset\skin-cancer-malignant-vs-benign\data\train\malignant"

dataset = SkinCancerDataset(benign_dir, malignant_dir, transform=transform)
dataloader = DataLoader(dataset, batch_size=32, shuffle=True)

print("Total number of images:", len(dataset))
```

Figure 4.10 Data Preprocessing Code Screenshot for Classifier Model (ResNet-18) 2

Based on Figures 4.9 and 4.10 shown, a custom PyTorch dataset was created to load the dataset from Kaggle, which is named “Skin Cancer: Malignant vs Benign”. This dataset included benign and malignant skin lesion images stored in different folders. Each of the images is labelled as malignant (1) or benign (0). The preprocessing pipeline has resized all the images to 224x224 pixels, coverts them to tensors, and used ImageNet mean and standard deviation to normalized them. A DataLoader was also implemented to load the dataset in 32 batches with shuffling enabled, to ensure the training of the classification model.

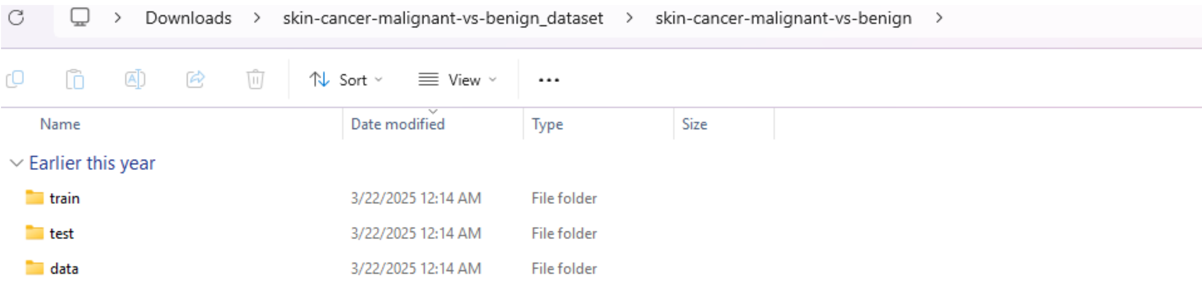

Figure 4.11 Dataset that used to trained classifier models

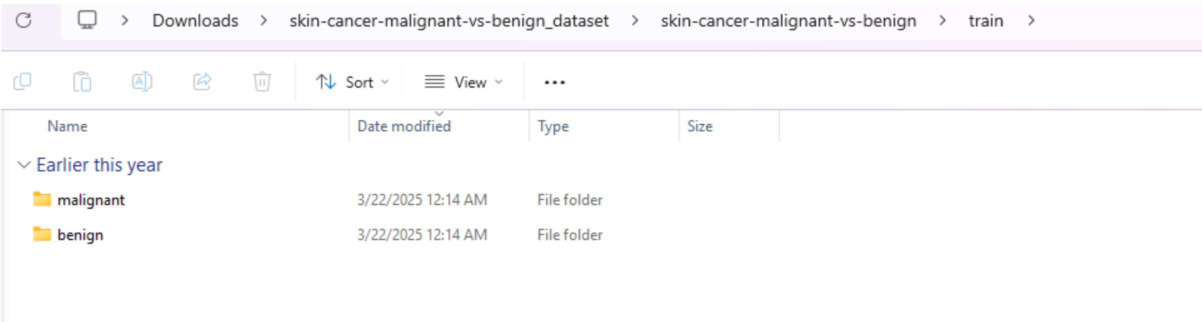

Figure 4.12 Dataset that used to trained classifier models in the test, train and data folder

Based on the Figure 4.11 shown, it was the dataset that used to trained the classifier model which include train, test and data. All of the folder will consist two separate folder that stored benign and malignant skin lesion images.

4.5 Model Training and Tuning

4.5.1 Model Training and Tuning for Segmentation Models (DeepLabV3 and Custom U-Net)

```
Epoch:  1 | Train mode      | Cost: 0.370 | Dice: 0.598: 100%|██████████| 130/130 [03:11<00:00,  1.47s/it]
          | Validation mode | Cost: 0.451 | Dice: 0.625: 100%|██████████| 44/44 [00:59<00:00,  1.35s/it]
Epoch:  2 | Train mode      | Cost: 0.262 | Dice: 0.733: 100%|██████████| 130/130 [03:18<00:00,  1.53s/it]
          | Validation mode | Cost: 0.233 | Dice: 0.776: 100%|██████████| 44/44 [00:59<00:00,  1.35s/it]
Epoch:  3 | Train mode      | Cost: 0.248 | Dice: 0.748: 100%|██████████| 130/130 [03:09<00:00,  1.45s/it]
          | Validation mode | Cost: 0.238 | Dice: 0.773: 100%|██████████| 44/44 [00:58<00:00,  1.34s/it]
Epoch:  4 | Train mode      | Cost: 0.225 | Dice: 0.778: 100%|██████████| 130/130 [03:13<00:00,  1.49s/it]
          | Validation mode | Cost: 0.218 | Dice: 0.800: 100%|██████████| 44/44 [00:59<00:00,  1.34s/it]
Epoch:  5 | Train mode      | Cost: 0.203 | Dice: 0.802: 100%|██████████| 130/130 [03:08<00:00,  1.45s/it]
          | Validation mode | Cost: 0.197 | Dice: 0.823: 100%|██████████| 44/44 [00:59<00:00,  1.34s/it]
Epoch:  6 | Train mode      | Cost: 0.197 | Dice: 0.802: 100%|██████████| 130/130 [03:13<00:00,  1.49s/it]
          | Validation mode | Cost: 0.196 | Dice: 0.824: 100%|██████████| 44/44 [00:59<00:00,  1.34s/it]
Epoch:  7 | Train mode      | Cost: 0.184 | Dice: 0.821: 100%|██████████| 130/130 [03:12<00:00,  1.48s/it]
          | Validation mode | Cost: 0.178 | Dice: 0.828: 100%|██████████| 44/44 [00:58<00:00,  1.33s/it]
Epoch:  8 | Train mode      | Cost: 0.172 | Dice: 0.831: 100%|██████████| 130/130 [03:16<00:00,  1.51s/it]
          | Validation mode | Cost: 0.172 | Dice: 0.832: 100%|██████████| 44/44 [00:58<00:00,  1.34s/it]
Epoch:  9 | Train mode      | Cost: 0.172 | Dice: 0.833: 100%|██████████| 130/130 [03:13<00:00,  1.49s/it]
          | Validation mode | Cost: 0.171 | Dice: 0.830: 100%|██████████| 44/44 [00:59<00:00,  1.34s/it]
Epoch: 10 | Train mode      | Cost: 0.170 | Dice: 0.831: 100%|██████████| 130/130 [03:07<00:00,  1.44s/it]
          | Validation mode | Cost: 0.180 | Dice: 0.817: 100%|██████████| 44/44 [00:58<00:00,  1.33s/it]
Epoch: 11 | Train mode      | Cost: 0.169 | Dice: 0.833: 100%|██████████| 130/130 [03:07<00:00,  1.45s/it]
          | Validation mode | Cost: 0.171 | Dice: 0.830: 100%|██████████| 44/44 [00:58<00:00,  1.32s/it]
Epoch: 12 | Train mode      | Cost: 0.166 | Dice: 0.832: 100%|██████████| 130/130 [03:07<00:00,  1.44s/it]
          | Validation mode | Cost: 0.167 | Dice: 0.833: 100%|██████████| 44/44 [00:58<00:00,  1.32s/it]
Epoch: 13 | Train mode      | Cost: 0.163 | Dice: 0.836: 100%|██████████| 130/130 [03:11<00:00,  1.47s/it]
          | Validation mode | Cost: 0.167 | Dice: 0.836: 100%|██████████| 44/44 [00:58<00:00,  1.33s/it]
Epoch: 14 | Train mode      | Cost: 0.168 | Dice: 0.831: 100%|██████████| 130/130 [03:14<00:00,  1.50s/it]
          | Validation mode | Cost: 0.165 | Dice: 0.835: 100%|██████████| 44/44 [00:57<00:00,  1.31s/it]
Epoch: 15 | Train mode      | Cost: 0.161 | Dice: 0.841: 100%|██████████| 130/130 [03:08<00:00,  1.45s/it]
          | Validation mode | Cost: 0.162 | Dice: 0.846: 100%|██████████| 44/44 [00:57<00:00,  1.32s/it]
Epoch: 16 | Train mode      | Cost: 0.162 | Dice: 0.836: 100%|██████████| 130/130 [03:08<00:00,  1.45s/it]
          | Validation mode | Cost: 0.163 | Dice: 0.841: 100%|██████████| 44/44 [00:57<00:00,  1.31s/it]
Epoch: 17 | Train mode      | Cost: 0.164 | Dice: 0.832: 100%|██████████| 130/130 [03:03<00:00,  1.41s/it]
          | Validation mode | Cost: 0.163 | Dice: 0.840: 100%|██████████| 44/44 [00:58<00:00,  1.32s/it]
Epoch: 18 | Train mode      | Cost: 0.164 | Dice: 0.835: 100%|██████████| 130/130 [03:09<00:00,  1.46s/it]
          | Validation mode | Cost: 0.165 | Dice: 0.839: 100%|██████████| 44/44 [00:57<00:00,  1.32s/it]
Epoch: 19 | Train mode      | Cost: 0.160 | Dice: 0.838: 100%|██████████| 130/130 [03:07<00:00,  1.44s/it]
          | Validation mode | Cost: 0.163 | Dice: 0.841: 100%|██████████| 44/44 [00:57<00:00,  1.31s/it]
Epoch: 20 | Train mode      | Cost: 0.163 | Dice: 0.835: 100%|██████████| 130/130 [03:07<00:00,  1.44s/it]
          | Validation mode | Cost: 0.164 | Dice: 0.836: 100%|██████████| 44/44 [00:58<00:00,  1.32s/it]
```

Figure 4.13 Training and validation loss of Custom U-Net Model

Based on the Figure 4.13 shown, the U-Net model had trained 20 epochs. The training loss had increased consistently which decreased from 0.370 to 0.163, while the loss of the validation also decreased from 0.451 to 0.164. The Dice coefficient had been improved from 0.598 at the first epochs to 0.851 on the training set and 0.836 at the validation set by epochs 20. These results have shown that the model have successfully converged without significant overfitting, and achieving a high accuracy of over 83% of the Dice score on the validation set.

```
Epoch:  1 | Train mode      | Cost: 0.222 | Dice: 0.797: 100%|██████████| 130/130 [03:10<00:00,  1.47s/it]
          | Validation mode | Cost: 0.186 | Dice: 0.825: 100%|██████████| 44/44 [00:58<00:00,  1.34s/it]
Epoch:  2 | Train mode      | Cost: 0.177 | Dice: 0.829: 100%|██████████| 130/130 [03:10<00:00,  1.46s/it]
          | Validation mode | Cost: 0.174 | Dice: 0.825: 100%|██████████| 44/44 [00:59<00:00,  1.34s/it]
Epoch:  3 | Train mode      | Cost: 0.169 | Dice: 0.836: 100%|██████████| 130/130 [03:08<00:00,  1.45s/it]
          | Validation mode | Cost: 0.155 | Dice: 0.850: 100%|██████████| 44/44 [00:58<00:00,  1.33s/it]
Epoch:  4 | Train mode      | Cost: 0.160 | Dice: 0.838: 100%|██████████| 130/130 [03:07<00:00,  1.45s/it]
          | Validation mode | Cost: 0.150 | Dice: 0.857: 100%|██████████| 44/44 [00:58<00:00,  1.33s/it]
Epoch:  5 | Train mode      | Cost: 0.151 | Dice: 0.848: 100%|██████████| 130/130 [03:09<00:00,  1.45s/it]
          | Validation mode | Cost: 0.154 | Dice: 0.850: 100%|██████████| 44/44 [00:58<00:00,  1.34s/it]
Epoch:  6 | Train mode      | Cost: 0.144 | Dice: 0.855: 100%|██████████| 130/130 [03:13<00:00,  1.49s/it]
          | Validation mode | Cost: 0.151 | Dice: 0.858: 100%|██████████| 44/44 [00:58<00:00,  1.34s/it]
Epoch:  7 | Train mode      | Cost: 0.141 | Dice: 0.861: 100%|██████████| 130/130 [03:07<00:00,  1.44s/it]
          | Validation mode | Cost: 0.146 | Dice: 0.861: 100%|██████████| 44/44 [00:59<00:00,  1.35s/it]
Epoch:  8 | Train mode      | Cost: 0.140 | Dice: 0.863: 100%|██████████| 130/130 [03:06<00:00,  1.44s/it]
          | Validation mode | Cost: 0.139 | Dice: 0.867: 100%|██████████| 44/44 [00:59<00:00,  1.34s/it]
Epoch:  9 | Train mode      | Cost: 0.134 | Dice: 0.865: 100%|██████████| 130/130 [03:08<00:00,  1.45s/it]
          | Validation mode | Cost: 0.145 | Dice: 0.857: 100%|██████████| 44/44 [00:59<00:00,  1.36s/it]
Epoch: 10 | Train mode      | Cost: 0.130 | Dice: 0.869: 100%|██████████| 130/130 [03:07<00:00,  1.44s/it]
          | Validation mode | Cost: 0.136 | Dice: 0.870: 100%|██████████| 44/44 [00:58<00:00,  1.33s/it]
Epoch: 11 | Train mode      | Cost: 0.131 | Dice: 0.869: 100%|██████████| 130/130 [03:09<00:00,  1.46s/it]
          | Validation mode | Cost: 0.135 | Dice: 0.871: 100%|██████████| 44/44 [00:58<00:00,  1.34s/it]
Epoch: 12 | Train mode      | Cost: 0.130 | Dice: 0.870: 100%|██████████| 130/130 [03:12<00:00,  1.48s/it]
          | Validation mode | Cost: 0.138 | Dice: 0.868: 100%|██████████| 44/44 [00:59<00:00,  1.35s/it]
Epoch: 13 | Train mode      | Cost: 0.134 | Dice: 0.872: 100%|██████████| 130/130 [03:13<00:00,  1.49s/it]
          | Validation mode | Cost: 0.139 | Dice: 0.867: 100%|██████████| 44/44 [00:59<00:00,  1.36s/it]
Epoch: 14 | Train mode      | Cost: 0.128 | Dice: 0.870: 100%|██████████| 130/130 [03:04<00:00,  1.42s/it]
          | Validation mode | Cost: 0.134 | Dice: 0.872: 100%|██████████| 44/44 [00:58<00:00,  1.32s/it]
Epoch: 15 | Train mode      | Cost: 0.130 | Dice: 0.871: 100%|██████████| 130/130 [03:06<00:00,  1.43s/it]
          | Validation mode | Cost: 0.132 | Dice: 0.873: 100%|██████████| 44/44 [00:58<00:00,  1.33s/it]
Epoch: 16 | Train mode      | Cost: 0.131 | Dice: 0.868: 100%|██████████| 130/130 [03:12<00:00,  1.48s/it]
          | Validation mode | Cost: 0.134 | Dice: 0.873: 100%|██████████| 44/44 [00:58<00:00,  1.34s/it]
Epoch: 17 | Train mode      | Cost: 0.129 | Dice: 0.873: 100%|██████████| 130/130 [03:06<00:00,  1.43s/it]
          | Validation mode | Cost: 0.139 | Dice: 0.864: 100%|██████████| 44/44 [00:59<00:00,  1.36s/it]
Epoch: 18 | Train mode      | Cost: 0.127 | Dice: 0.871: 100%|██████████| 130/130 [03:07<00:00,  1.44s/it]
          | Validation mode | Cost: 0.134 | Dice: 0.873: 100%|██████████| 44/44 [00:59<00:00,  1.36s/it]
Epoch: 19 | Train mode      | Cost: 0.131 | Dice: 0.869: 100%|██████████| 130/130 [03:08<00:00,  1.45s/it]
          | Validation mode | Cost: 0.135 | Dice: 0.871: 100%|██████████| 44/44 [00:59<00:00,  1.35s/it]
Epoch: 20 | Train mode      | Cost: 0.126 | Dice: 0.874: 100%|██████████| 130/130 [03:09<00:00,  1.46s/it]
          | Validation mode | Cost: 0.134 | Dice: 0.871: 100%|██████████| 44/44 [00:59<00:00,  1.34s/it]
```

Figure 4.14 Training and validation loss of DeepLabV3 Model

Based on the Figure 4.14 shown, the DeepLabV3 model was trained 20 epochs. The training loss decreased consistently from 0.222 to 0.126, while the loss of the validation had decreased from 0.186 to 0.134. The Dice score had increased from 0.797 at the first epochs to 0.874 on the training set and 0.871 at the validation set by epochs 20. In the comparison between Custom U-Net and DeepLabV3, DeepLabV3 have achieve a better segmentation performance because it has higher accuracy compared to Custom U-Net which DeepLabV3 had achieve over 87% of the accuracy while Custom U-Net only achieve over 83% of accuracy.

4.5.2 Model Training and Tuning for Classifier Models (ResNet-18)

```
Epoch 1/10: 100%|██████████| 83/83 [09:49<00:00,  7.10s/it, loss=0.136]
Epoch 1/10, Loss: 0.4099
Epoch 2/10: 100%|██████████| 83/83 [09:39<00:00,  6.99s/it, loss=0.524]
Epoch 2/10, Loss: 0.3346
Epoch 3/10: 100%|██████████| 83/83 [09:37<00:00,  6.95s/it, loss=0.268]
Epoch 3/10, Loss: 0.2998
Epoch 4/10: 100%|██████████| 83/83 [09:38<00:00,  6.97s/it, loss=0.659]
Epoch 4/10, Loss: 0.2739
Epoch 5/10: 100%|██████████| 83/83 [09:38<00:00,  6.96s/it, loss=0.527]
Epoch 5/10, Loss: 0.2607
Epoch 6/10: 100%|██████████| 83/83 [09:35<00:00,  6.93s/it, loss=0.438]
Epoch 6/10, Loss: 0.2379
Epoch 7/10: 100%|██████████| 83/83 [09:36<00:00,  6.95s/it, loss=0.0723]
Epoch 7/10, Loss: 0.2046
Epoch 8/10: 100%|██████████| 83/83 [09:51<00:00,  7.13s/it, loss=0.214]
Epoch 8/10, Loss: 0.2080
Epoch 9/10: 100%|██████████| 83/83 [11:22<00:00,  8.22s/it, loss=0.704]
Epoch 9/10, Loss: 0.1700
Epoch 10/10: 100%|██████████| 83/83 [11:21<00:00,  8.21s/it, loss=0.444]
Epoch 10/10, Loss: 0.1517
```

Figure 4.15 Training Loss of ResNet-18 Model

Based on the Figure 4.15 shown, the training process of the classifier model which is ResNet-18 classifier over 10 epochs. The training loss have decreased steadily from 0.4099 to 0.1517, which shows that the model had successfully learned how to differentiate between benign and malignant skin lesion.

4.6 Performance Evaluation of the Model

4.6.1 Performance Evaluation of Segmentation Model (DeepLabV3 and Custom U-Net)

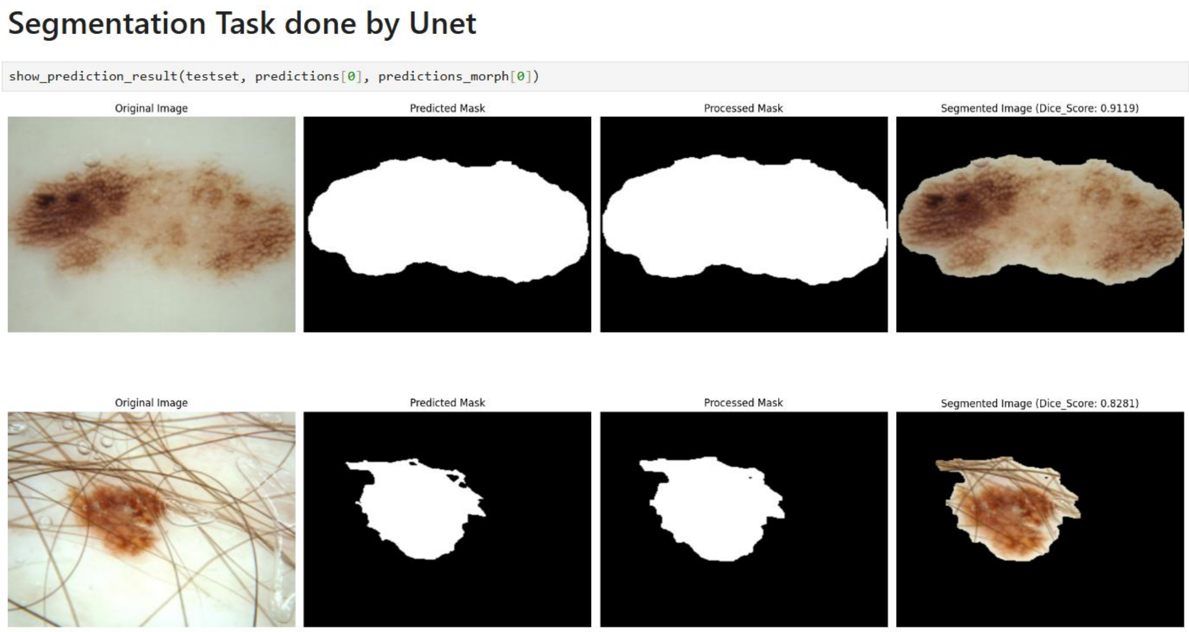

Figure 4.16 Dice score of Custom U-Net

Based on Figure 4.16 shown, the performance evaluation of Custom U-Net which is one of the segmentation models in the project was achieve a high Dice score which more than 0.8. For instance, the figure shown that the each of the test example, the original skin lesion image, the raw predicted mask, the refined of the morphological mask after processing, and the final segmented images are displayed. The Dice score for each image have shown how closely the predicted mask matches the ground truth such as Custom U-Net achieved Dice scores of 0.9119 and 0.8281 for two of the examples, which proved that this segmentation model has a strong segmentation performance, although the hair at the second images have slightly affect the accuracy of the model but still remain a high Dice score.

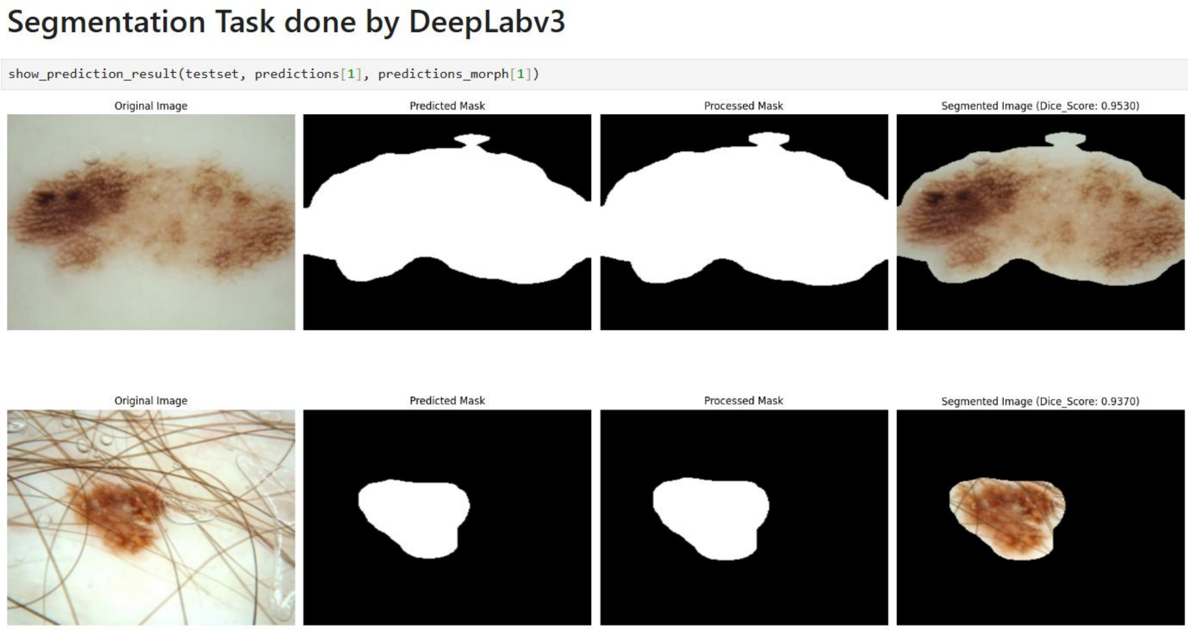

Figure 4.17 Dice score of DeepLabV3

Based on Figure 4.17 shown, the performance evaluation of DeepLabV3 which is second segmentation models in the project also achieve a high Dice score which more than 0.9 at the average Dice score. For instance, each row shows the original skin lesion image, the raw predicted mask, the processed mask after the refinement of the morphological, and the final segmented lesion accompany with the Dice score. The model had achieved Dice score for first images at 0.9530 and 0.9370 for the second image, which are higher than the corresponding Custom U-Net results. This have shown that DeepLabV3 model also have a high accuracy on segmented the skin lesion image, in the challenging cases with hair and noise occlusion. The reason that using model in this project instead of just use the high accuracy one is both model were implemented in this project for comprehensive evaluation. For example, Custom U-Net is classical baseline that widely used in the medical image, while DeepLabV3 represents a more advanced models with more accurate performance. By combining these two models can benefit the automated system become a more flexibility and complete system.

4.6.1.1 Graphs for Performance Evaluation on Segmentation Model (DeepLabV3 and Custom U-Net)

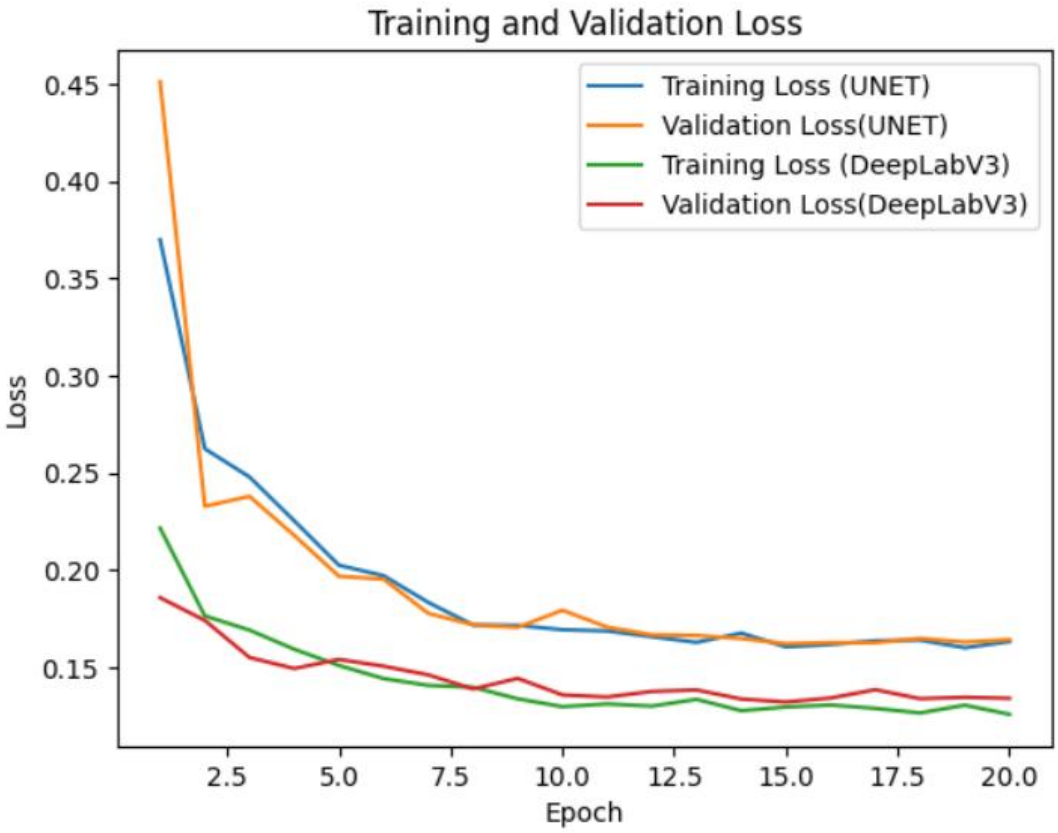

Figure 4.18 Training and Validation Loss Graphs of Segmentation Model (DeepLabV3 and Custom U-Net)

Based on the Figure 4.18 shown, is the training and validation loss of segmentation models in graph format. The figure show that both of the models present in decreasing loss values, indicating successful convergence. The final loss of Custom U-Net is approximately 0.16 to 0.17, while the DeepLabV3 had reached a lower loss compared to Custom U-Net which had loss around 0.12 to 0.13, which show that DeepLabV3 have a superior performance. Not only that, the training loss for both models have closely followed by the validation los, which suggest there are no significant overfitting occurred. Overall, DeepLabV3 model have a more accurate performance compared to Custom U-Net model. Sinc, the epochs are the average loss, therefore the decrease in consistently, while the graphs will be more detailed, so it will have slightly increased in the value.

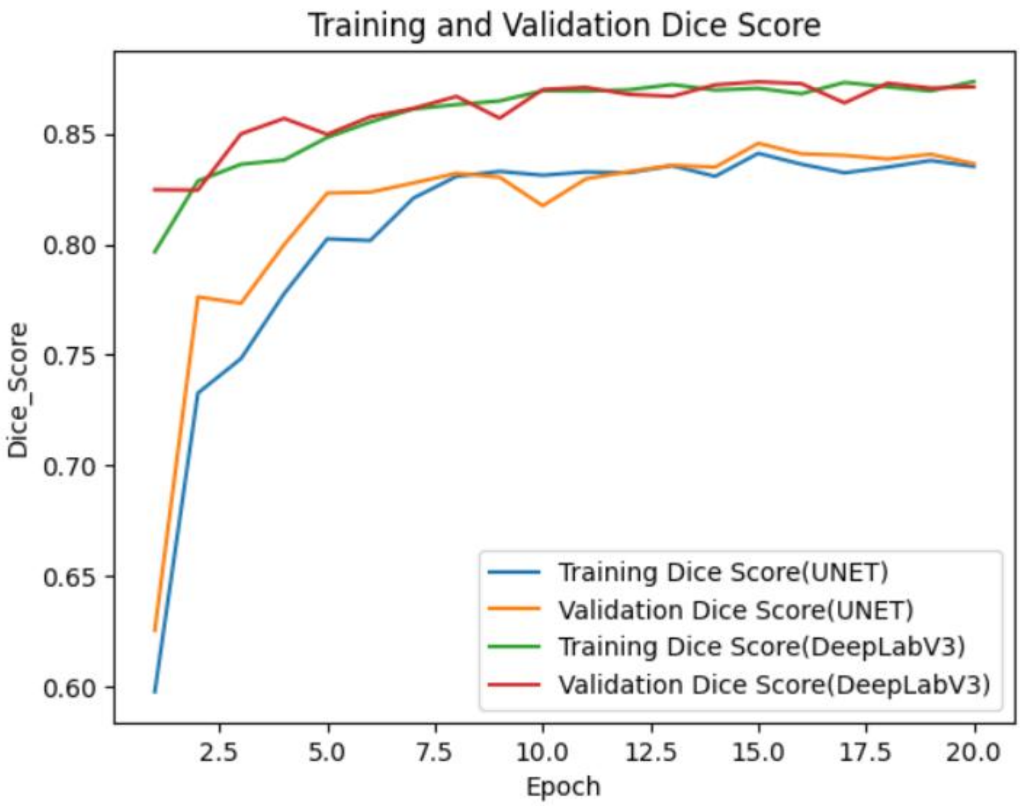

Figure 4.19 Training and Validation Dice Score of Segmentation Model (DeepLabV3 and Custom U-Net)

Based on the Figure 4.19 shown, the training and validation Dice score of the segmentation model had train over 20 epochs. In the figure, both of the models show a consistent increase in Dice score, indicate a successful learning. Custom U-Net reached a final Dice score of nearly 0.83, while DeepLabV3 achieved around 0.87. The training curves that followed closely by the validation curves, suggest a minimal overfitting. Therefore, DeepLabV3 model show a great performance compared to Custom U-Net model which have higher accuracy on segmentation task and faster convergence.

4.6.2 Performance Evaluation of Classifier Model (ResNet-18)

```python
def evaluate_accuracy(model, dataloader, device):
    model.eval()  # set to evaluation mode
    correct = 0
    total = 0
    with torch.no_grad():
        for inputs, labels in dataloader:
            inputs = inputs.to(device)
            labels = labels.to(device)
            outputs = model(inputs)  # raw logits from the model
            # Apply sigmoid to logits to get probabilities, then binarize with 0.5 threshold
            preds = (torch.sigmoid(outputs) > 0.5).float()
            # If preds shape does not match labels (e.g., extra dimensions), use squeeze()
            correct += (preds.squeeze() == labels.float()).sum().item()
            total += labels.size(0)
    accuracy = correct / total
    return accuracy

# Evaluate model accuracy:
accuracy = evaluate_accuracy(classifier_model, dataloader, device)
print("Accuracy: {:.2f}%".format(accuracy * 100))
```

```
Accuracy: 93.59%
```

Figure 4.20 Accuracy of ResNet-18

Based on the Figure 18 shown, it shows the accuracy of classifier model which is ResNet-18. Since, the training of the classifier model is simpler compared to the segmentation model, therefor the information that shown the performance are not detailed like the segmentation model but still can observe a good performance for the model. For instance, the loop of the evaluation applied in sigmoid activation followed by a 0.5 threshold to classify the lesion is benign or malignant. The model had achieved an accuracy of 93.59%, which proved that the classifier model had a good performance on classify the skin lesion type in a limited dataset. Although have high accuracy but it had overfitting while applied in the system, thus the threshold had been adjusted when applied in the automated system such as benign threshold is 0.15 while the malignant threshold is 0.85

4.7 Flask and Web Interface Deployment

The Flask will be deployed locally using the Microsoft Visual Studio Code, which also will integrate the trained deep learning models with a user-friendly web interface. The backend will be implemented in app.py file in the FYP2-Test folder, which will manage the file uploads, directories (static/results and static/uploads), and load the segmentation and classification pipeline that named in model.py. Initially, three model checkpoint and weight will be loaded inside the memory such as segmentation model (DeepLabV3 and Custom U-Net) to segmentate the skin lesion and classifier model (ResNet-18) to classify the skin lesion type (benign or malignant). When user upload a skin lesion image on web interface though the Flask, the images will be preprocessing to a size that able accept by the segmentation model, then the segmentation model will start to segmentate it, then the most reliable segmentate image will be pass to classifier model to do classification. The classifier will produce a diagnosis on detailed information such as probability of classification, original image, overlay image, skin lesion type and confidence level.

Furthermore, for the frond-end interface which is the web interface, the code will be stored in index.html in FY2-Folder. The web interface will be providing a simple interface for the user to upload the image. Then, user only need to wait less than one minute to received their result. To avoid the user put a blur image, the web interface also will give advice on the top of the web interface to convince the user upload at least 720p resolution image for processing. Additionally, since the result only for reference, therefore in the web interface also have state a caution of “This result is for reference only. If you notice any discomfort or suspicious change, please consult a qualified doctor.”.

Chapter 5

System Implementation

5.1 Hardware Setup

Figure 5.1 Laptop Setup

Based on Figure 5.1 shown, this model of laptop was used to implemented the segmentation model, classifier model and create the web interface for the user for interaction. This version of laptop is sufficient to stored the checkpoint and weight of the model and able to implemented all the model accompany with the web interface smoothly.

5.2 Software Setup

5.2.1 Anaconda

Figure 5.2 Anaconda

Based on the Figure 5.2 shown, it is Anaconda which play as a environment manager to make the library installation more simple such as PyTorch, Torchvision, OpenCV and Flask. This program also able to maintain the consistency of development and testing.

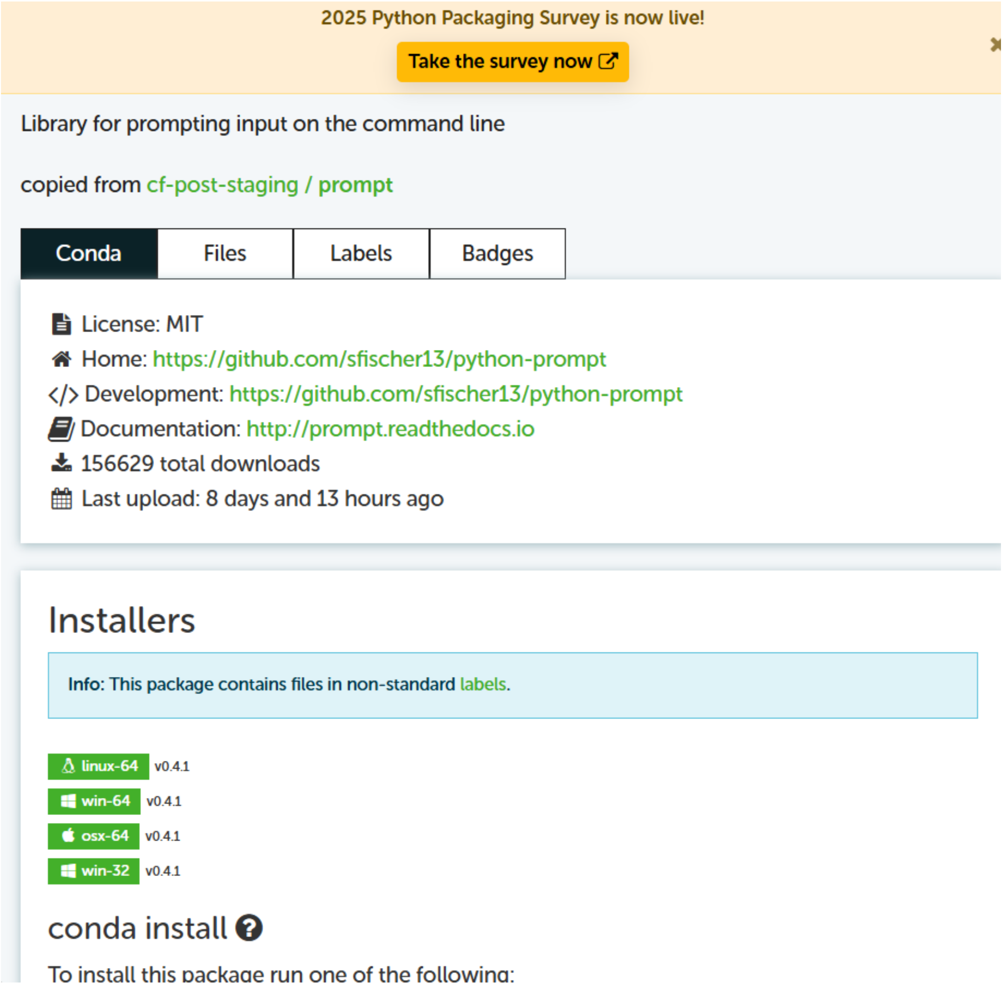

Figure 5.3 Anaconda Installation

Based on the Figure 5.3 shown, it was the Anaconda installation webpage, then install the Anaconda based on the version that suitable on the laptop. After the installation, Anaconda will be provided a stable environment on manage PyTorch, Torchvision, Flask and OpenCV to ensure the productivity of the experiment.

5.2.2 Jupyter Notebook

Figure 5.4 Jupyter Notebook

In this project, Jupyter Notebook had played a significant role for running the pre-trained AI model such as segmentation model (DeepLabV3 and Custom U-Net) and classifier model (ResNet-18). It was the coding program that widely used by the global, since it able to provide a detailed information to showing the result such as training and validation loss, epochs and so on. Other than that, it is suitable for the big AI model training.

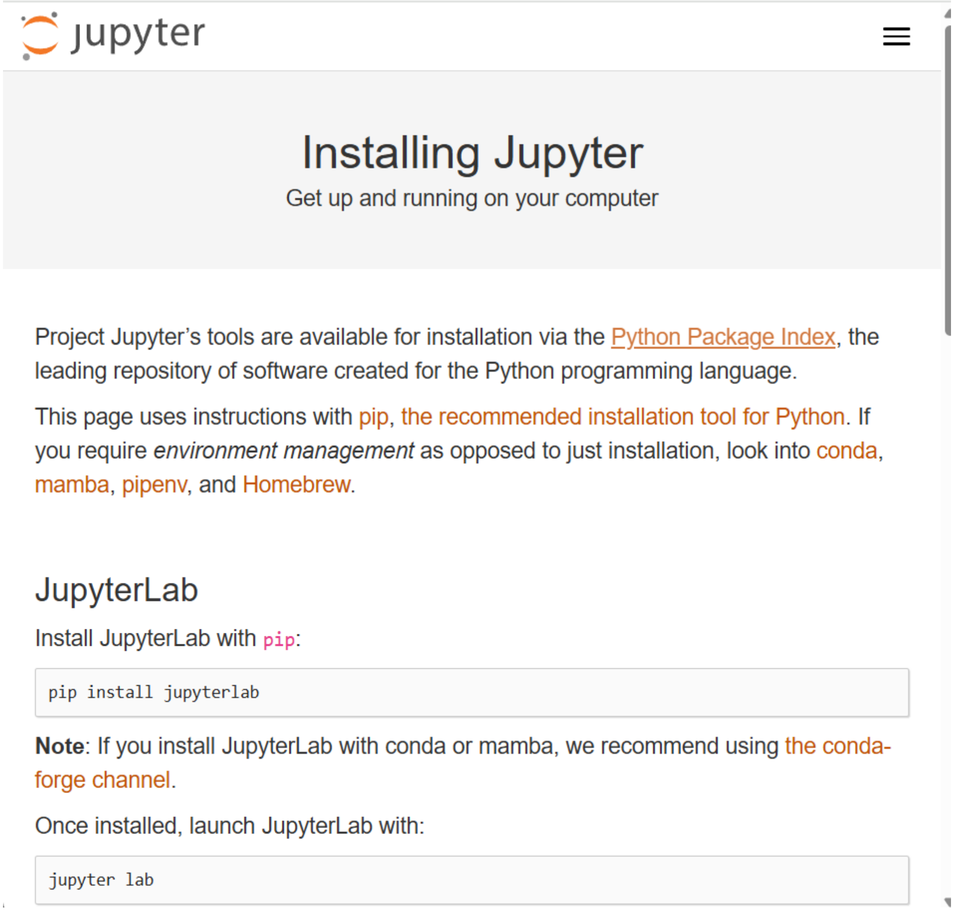

Figure 5.5 Jupyter Notebook Installation

Based on the Figure 5.3 shown, it was the website that guide the user on how to install the Jupyter Notebook. These tools need to install before proceeding to run the pre-trained segmentation model.

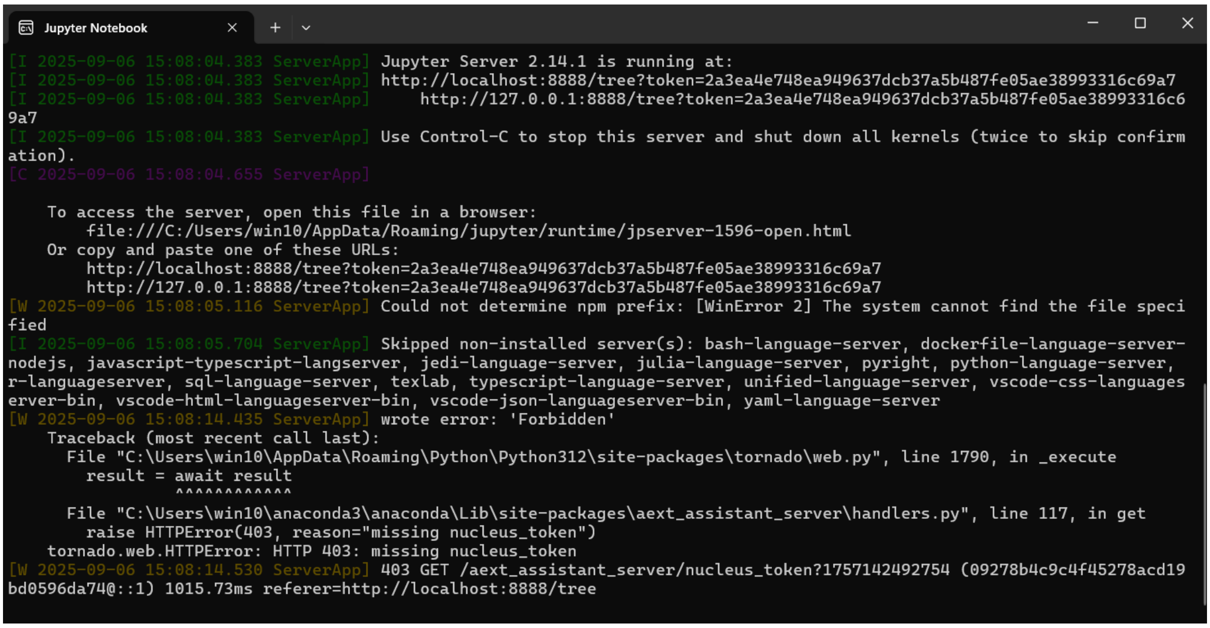

Figure 5.6 Launch Jupyter Notebook on Command Prompt

Based on the Figure 5.6 shown, after successfully install the Anaconda and Jupyter Notebook, then just direct click the Jupyter Notebook desktop icon, thus it will direct to the command prompt and initialize Anaconda to launch the Jupyter Notebook automatically.

5.2.3 Visual Studio Code

Figure 5.7 Visual Studio Code

Based on the Figure 5.7 shown, it was the Visual Studio Code. In this project, Visual Studio Code are handling on implemented all the pre-trained model together by load the weight and the checkpoint of segmentation model and classifier model. Not only than that, it also handling on building a Flask and web interface for the user which Flask able to manage the process image through backend and user able to upload the skin lesion image through frond end (web interface).

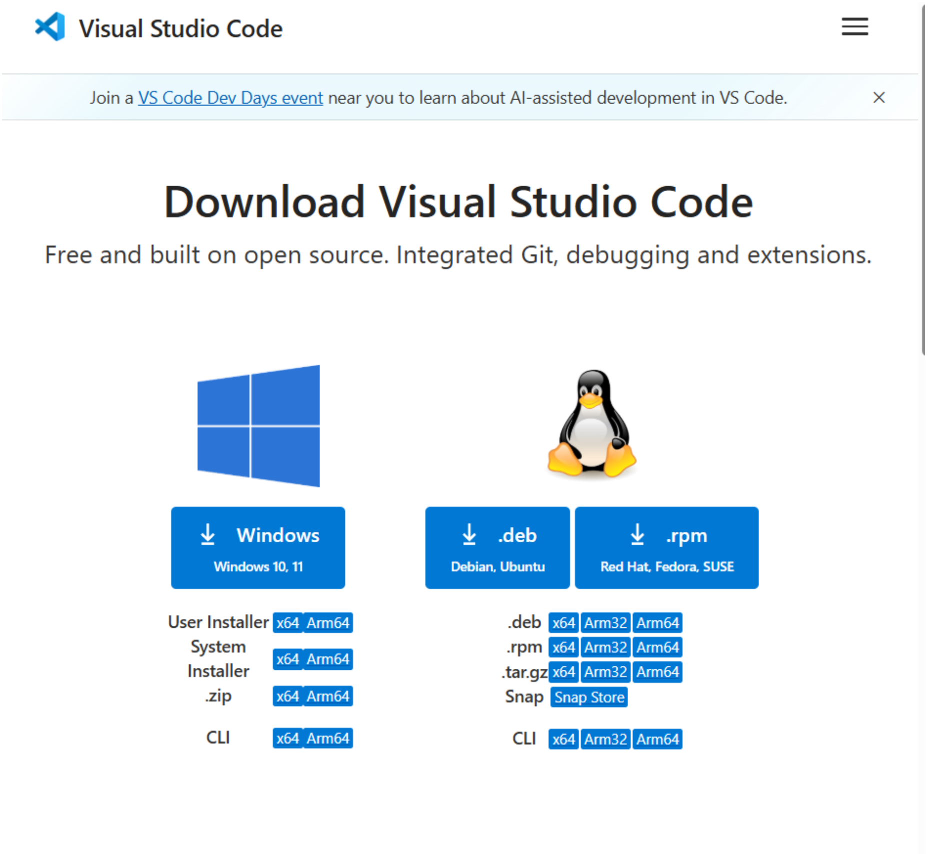

Figure 5.8 Visual Studio Code Installation

Install the Visual Studio Code through the official website and click the install Window button to install the Visual Studio Code

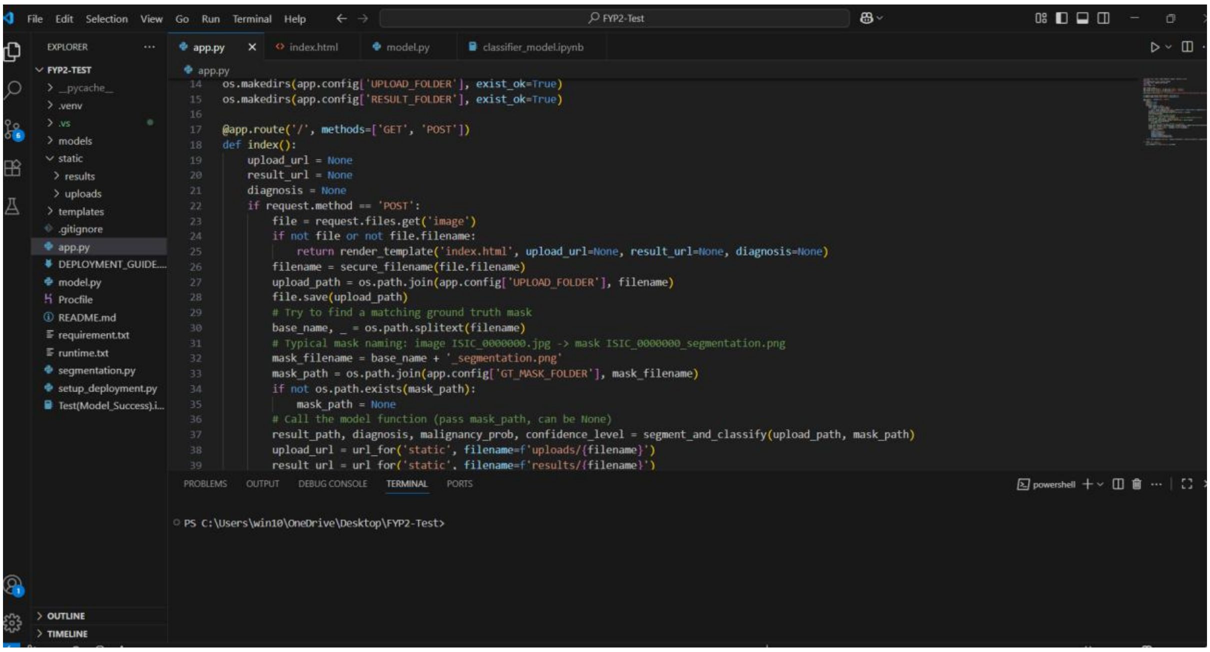

Figure 5.9 Interior View of Visual Studio Code

After successfully install Visual Studio Code, then will be able to choose a folder and launch the folder on the program.

5.3 Setting and Configuration

5.3.1 Jupyter Notebook Setting and Configuration

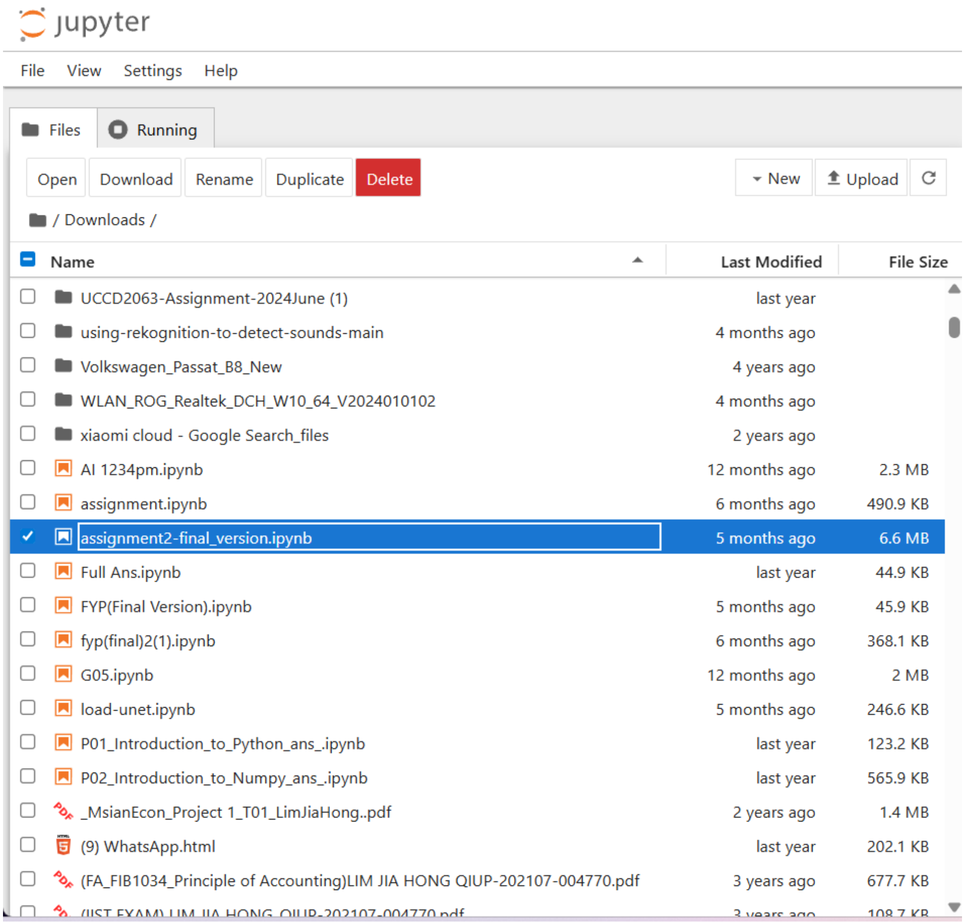

Figure 5.10 Open Pre-trained Segmentation Model File

Based on the Figure 5.10 shown, the file named "assignment2-final_version. ipynb" is the file that stored the pre-trained code of the segmentation model (DeepLabV3 and Custom U-Net). After launch the Jupyter Notebook, need to click on this file to run the pre-trained code to store the checkpoint and the weight of the segmentation model.

Group 8: Image Segmentation of Lesion on Human Skin

Coded by All: Kong Wai Kin, Adele Lim Hui Hui, Boey Hou Yan, Wai Jia Le

[2]:
```
import os
import glob
import tarfile
import random
import shutil

from collections import defaultdict
from tqdm import tqdm
import numpy as np
import matplotlib.pyplot as plt
from PIL import Image

import torch
from torch import nn
from torch.nn import functional as F
from torch.utils.data import Dataset
from torch.utils.data import DataLoader
from torch.utils.data import Subset

from torchvision.transforms import v2
from torchvision.transforms import InterpolationMode
from torchvision import tv_tensors #every do with tensor , mask/bounding box/image differentiate them, depend type of tensors, transformation

from torchinfo import summary
```

[4]:
```
#Specify the Device, if gpu is not found then use CPU
DEVICE = torch.device("cuda:0" if torch.cuda.is_available() else "cpu")

#state the global variables for learnin rate, batch size, epochs, mean and standard deviation before training any models
LR = 0.001
BATCH_SIZE = 12
NUM_EPOCHS = 20
MEAN = (0.485, 0.456, 0.406)
STD  = (0.229, 0.224, 0.225)
```

5.11 Interior View of the Segmentation Model Code

Based on the Figure 5.11, shown it was the interior view of the “assignment2-final_version. ipynb” which is the segmentation model. This code is done by student from FICT of Universiti Tunku Abdul Rahman [10]. This were their assignment on a title of “Image Segmentation of Lesion on Human Skin”. This model had been applied on this automated system to achieve a consistency of the system.

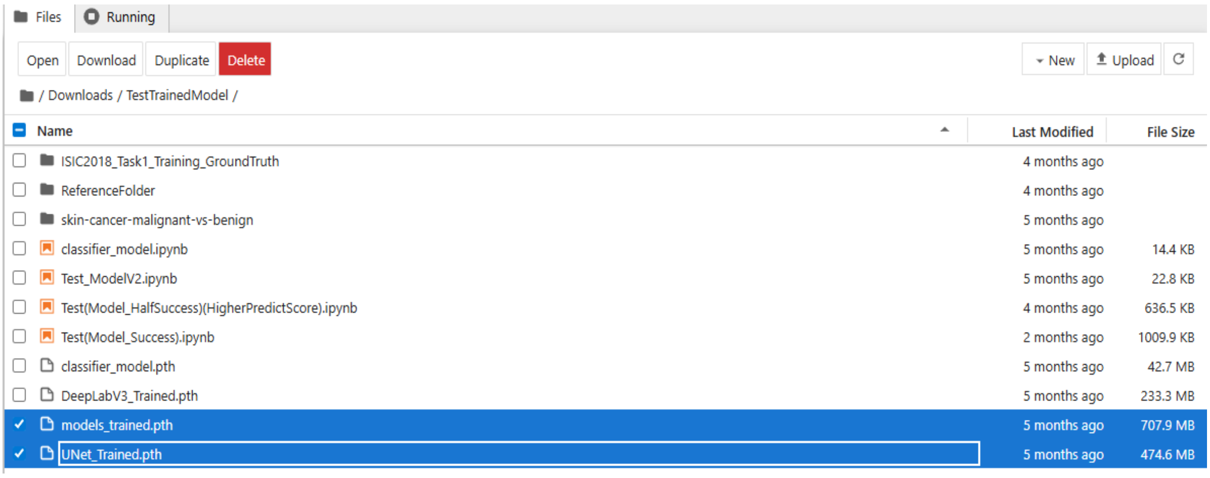

Figure 5.12 File that Store the Checkpoint and Weight of Segmentation Model (DeepLabV3 and Custom U-Net)

Based on the Figure 5.12 shown, it was the file that store the checkpoint and weight of segmentation model after successfully run the code in “assignment2-final_version. ipynb” file. The checkpoint and weight of the model will be export to two files, for instance DeepLabV3 model will be store in the file named “model_trained.pth” and Custom U-Net model will be stored in “UNet_Trained.pth”. The reason that used “model_trained.pth” instead of use “DeepLabV3_Trained.pth” because the “DeepLabV3_Trained.pth” does not stored the checkpoint and weight in a proper way, therefore re process the code again and store the DeepLabV3 model in “model_trained.pth” file.

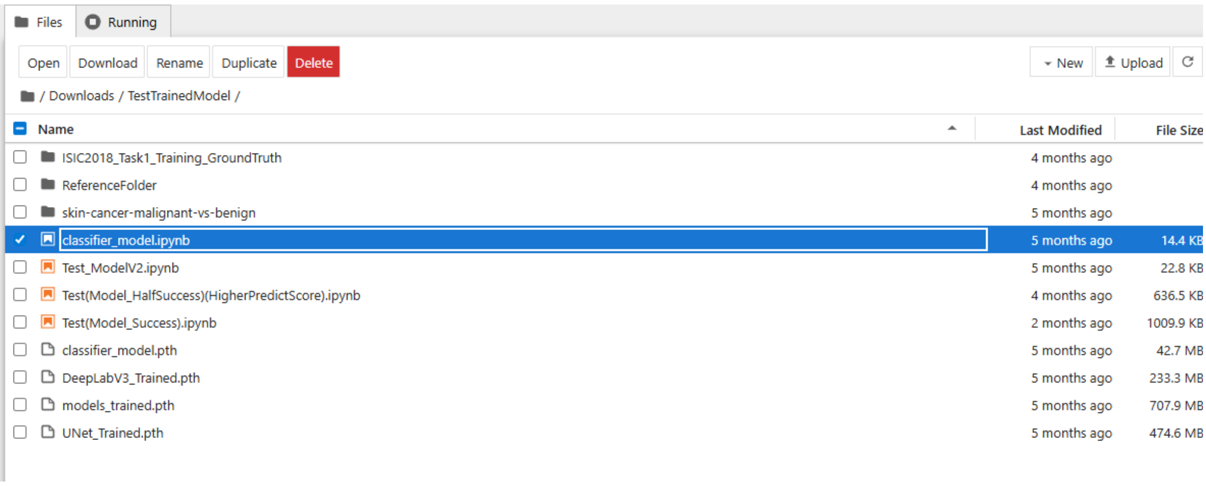

Figure 5.13 Open the Pre-trained Classifier Model File

Based on the Figure 5.13 shown, is open the pre-trained classifier model (ResNet-18) file. The classifier model code is stored in "classifier_model. ipynb" file. After launch the Jupyter Notebook, need to click on this file to run the pre-trained code to store the checkpoint and the weight of the classifier model.

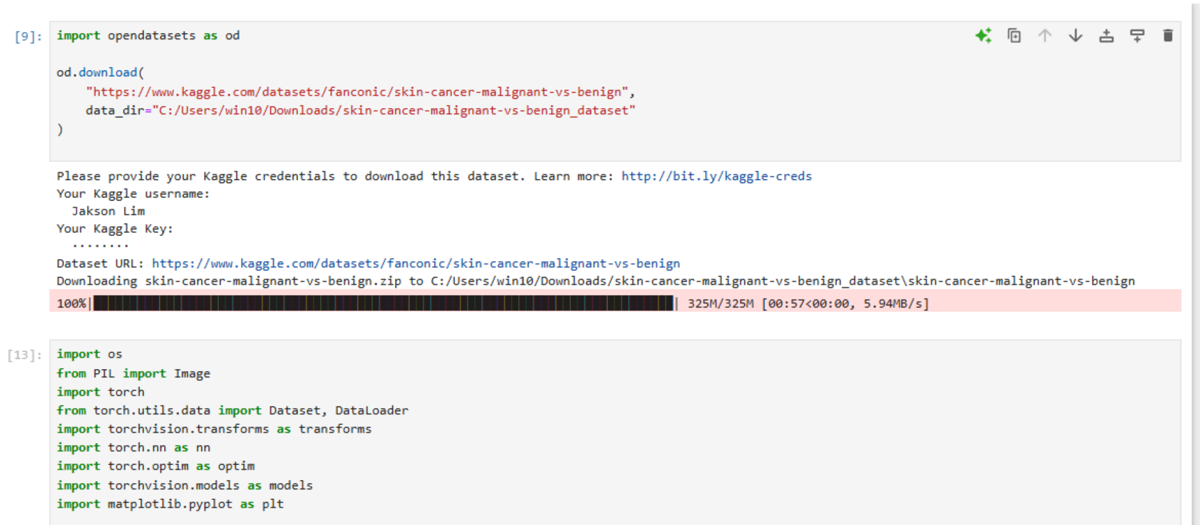

Figure 5.14 Interior View of the Classifier Model (ResNet-18)

Based on the Figure 5.14 shown, it was the code inside the "classifier_model. ipynb" file. After run the code, need to access to Kaggle account and get the dataset for classifier model training because the dataset is from the other author, therefore need to get from him to run the pre-trained classifier model.

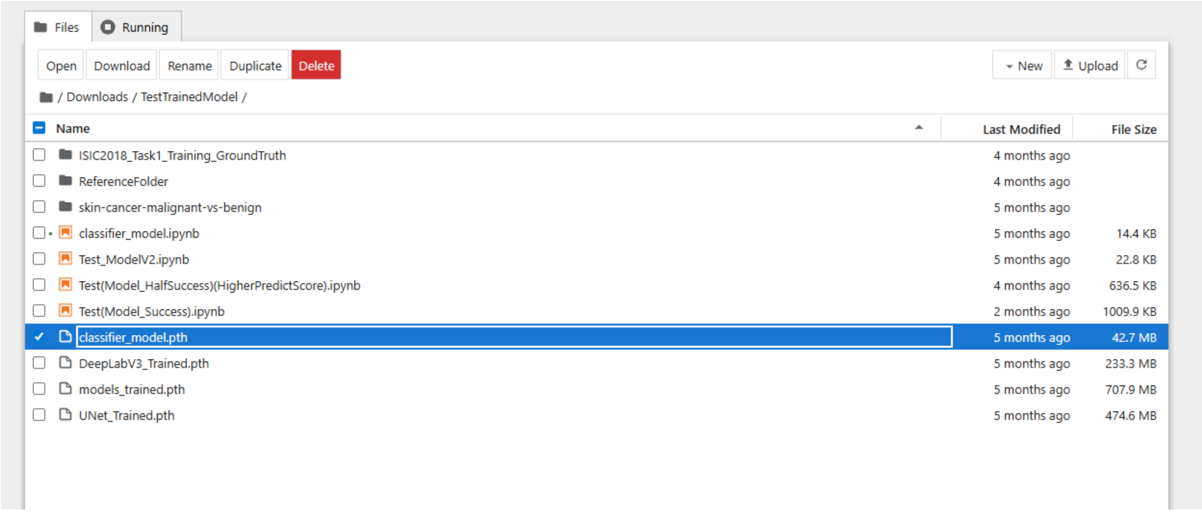

Figure 5.15 File that Store the Checkpoint and Weight of Classifier Model (ResNet-18)

Based on the Figure 5.15 shown, it was the file that stored the checkpoint and the weight of the ResNet-18 model. After successfully run the pre-trained code, the checkpoint and the weight of the classifier model will be stored in a file named "classifier_model.pth".

5.3.2 Visual Studio Code Setting and Configuration

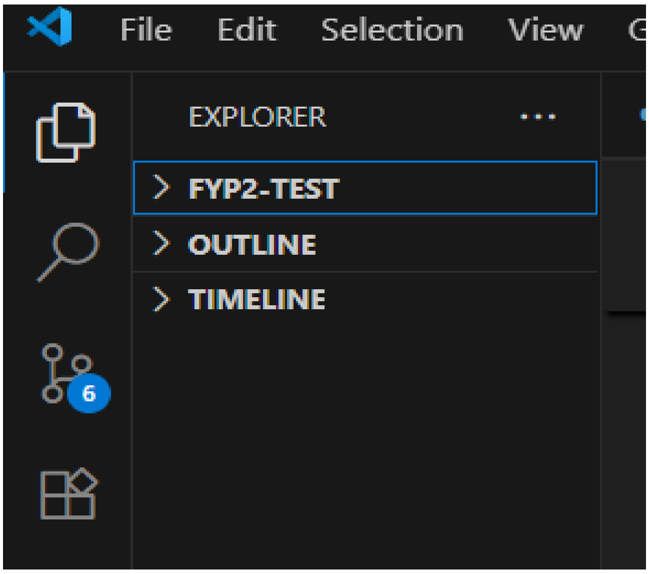

Figure 5.16 Folder that Stored the Content Needed for the Automated System

Based on the Figure 5.16 shown, is create a folder to store all the required content for the automated system. For instance, this project has created a folder named "FYP2-Test" to stored the flask, web interface and load the checkpoint and weight of the model to ensure the automated system able to run smoothly.

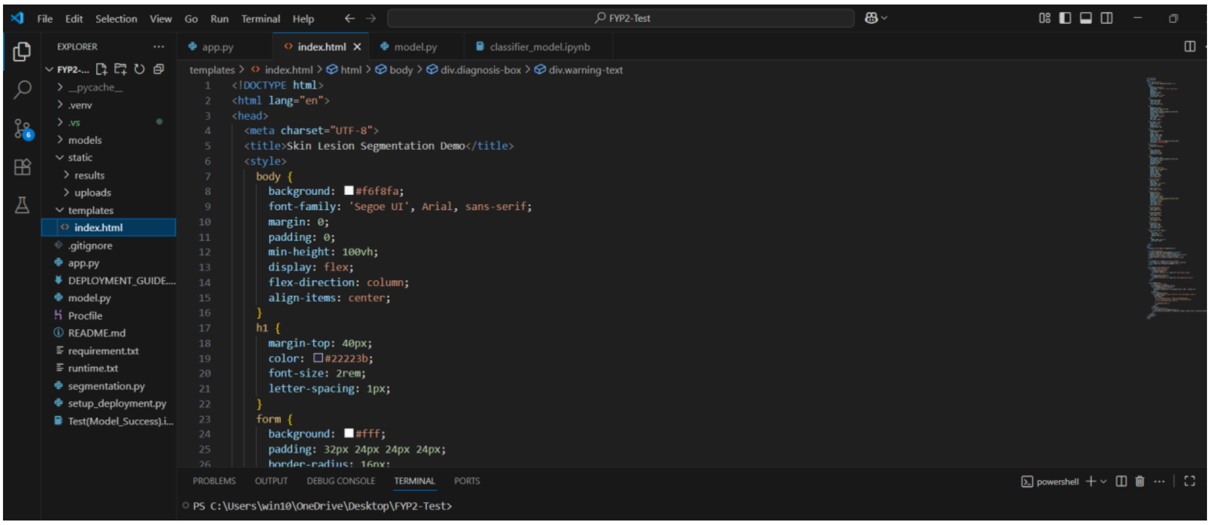

Figure 5.17 File that Stored the Template of the Web Interface

Based on the Figure 5.17 shown, it was the file that stored the template of the web interface. The template of the website is in a file named “index.html”.

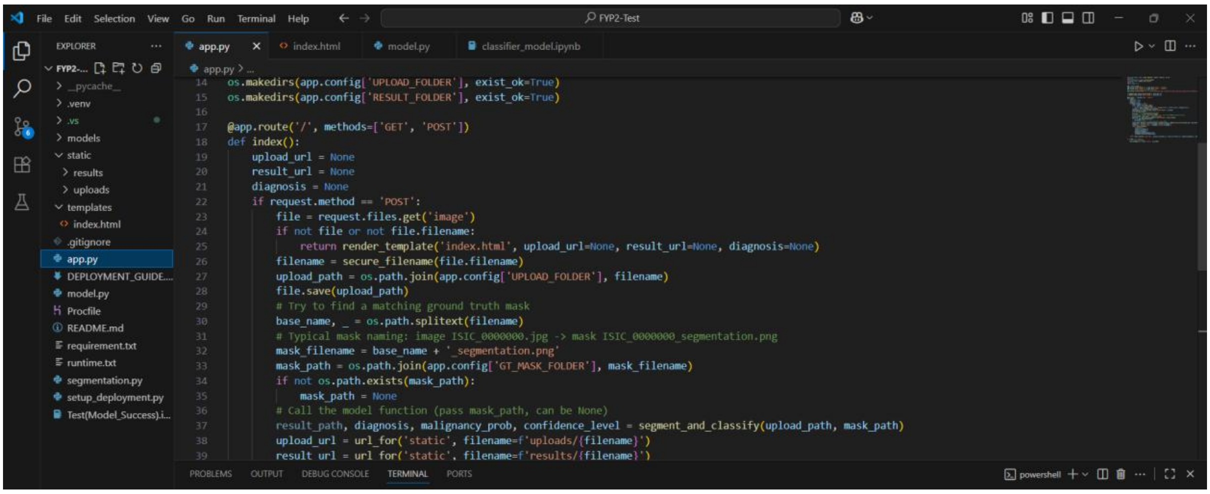

Figure 5.18 File that Stored the Flask code

Based on the Figure 5.18 shown, it was the code of Flask which play a very significant role in this project without it, the automated system will not able to work. The code of Flask is stored in a file named “app.py”. Before launch the automated system, run this file will be start to launch the system.

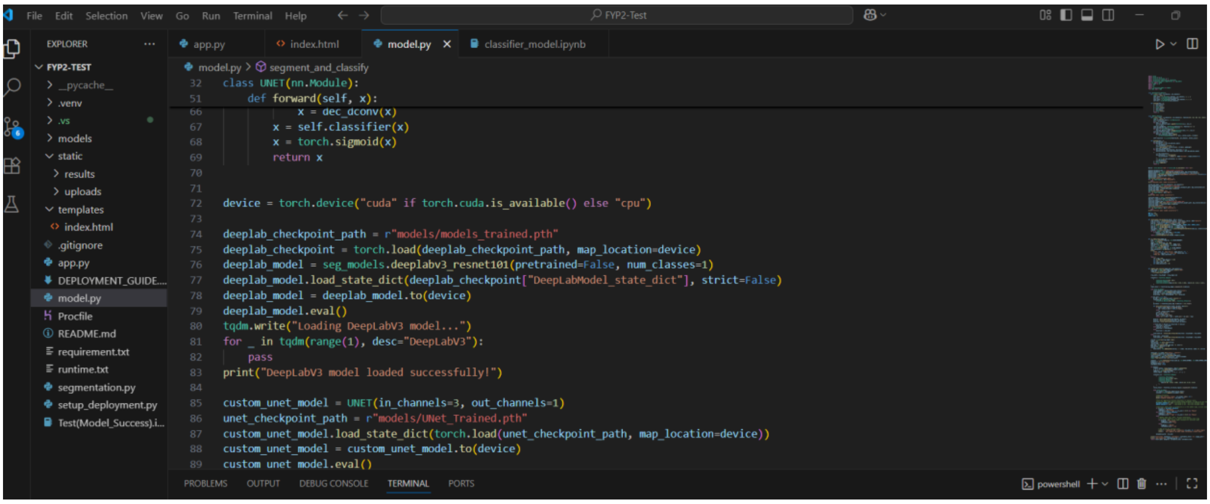

Figure 5.19 File that Uses to Load the Checkpoint and Weight of Segmentation Model and Classifier Model

Based on the Figure 5.19 shown, it was the file that used to load the checkpoint and weight of the segmentation model and classifier model. The file that used to load is “model.py. This file does not need to run separately just run the “app.py” file this file will be run automatically.

5.4 System Operations (with screenshot)

5.4.1 Output of Result shown by Segmentation Model (example image excluded from the test data and train data)

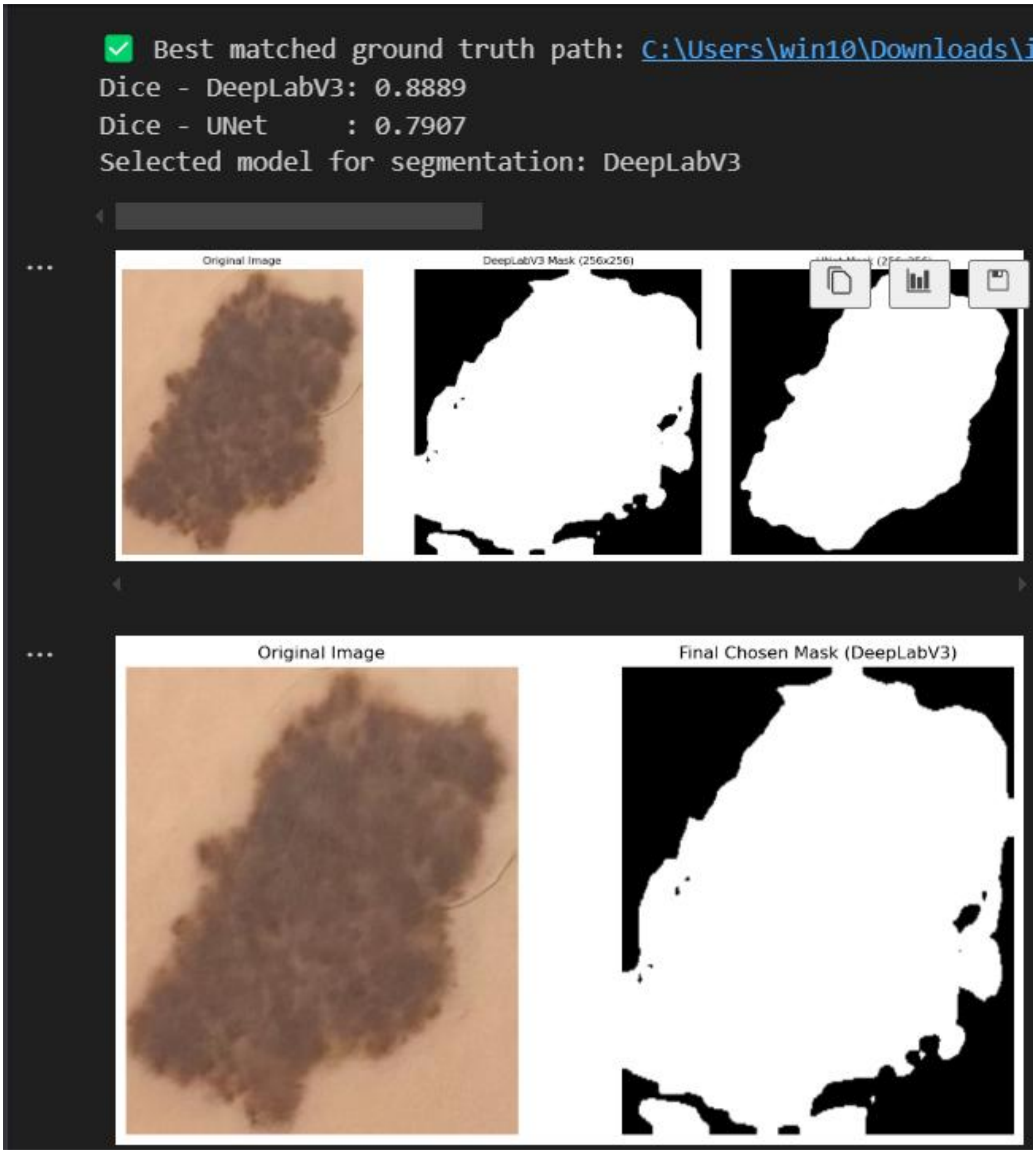

Figure 5.20 Segmentation result done by DeepLabV3 and Custom U-Net

Based on the Figure 5.20 shown, the segmentation model able to segmentate the image that excluded from the train and test dataset in an accuracy probability which having a 0.7924 which is (79.24%) of dice on DeepLabV3 model and 0.7907 which is (79.07%) of dice on Custom U-Net model. Due to this incident, the model has a high dice will be choose, then DeepLabV3 had been chosen.

5.4.2 Output of Result shown by Classifier Model (example image excluded from the test data and train data)

Figure 5.21 Classification probability and final diagnosis that done by ResNet-18 classifier model

Based on Figure 5.21 shown, the classification probability on the image that exclude from the test and train data had a 17.53% which lower than 50% of the probability to become malignant. Therefore, the final diagnosis will be benign. This is the classifier model that before setting the threshold value.

5.4.3 Output of Result shown on Web Interface (example image excluded from the test data and train data)

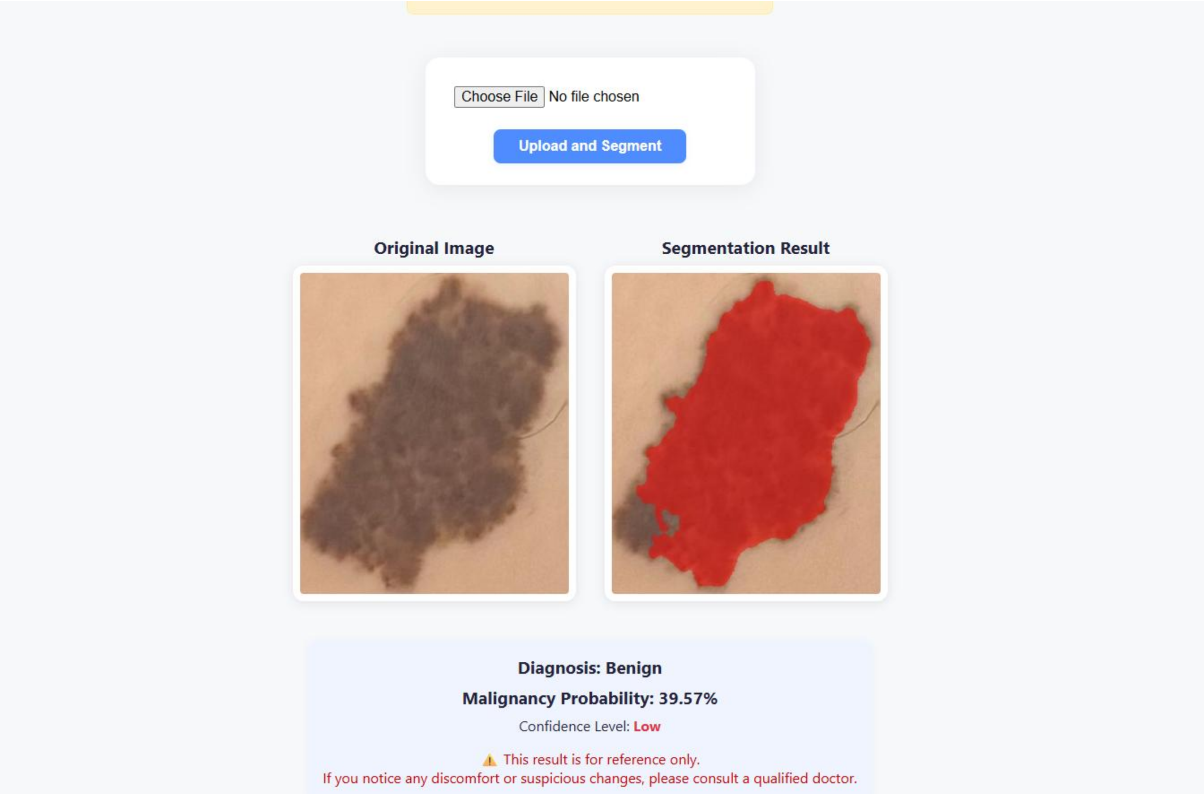

Figure 5.22 Result shown on Web Interface

Based on the Figure 5.22 shown, the diagnosis result is benign but the malignancy probability is higher than the previous testing because the threshold before is overfitting, therefor to adjust the threshold to overcome this problem. Furthermore, also add a confidence level to tell the user how confidence of the model on classify it is benign or malignant. For example, this probability over the confidence level of medium and high of benign, so it is low but it is still not considered as a malignant skin. Not only than that, also state a caution which is “The result is for reference only. If you notice any discomfort or suspicious changes, please consult a qualified doctor.”.

5.4.4 Main Page of the Automated System

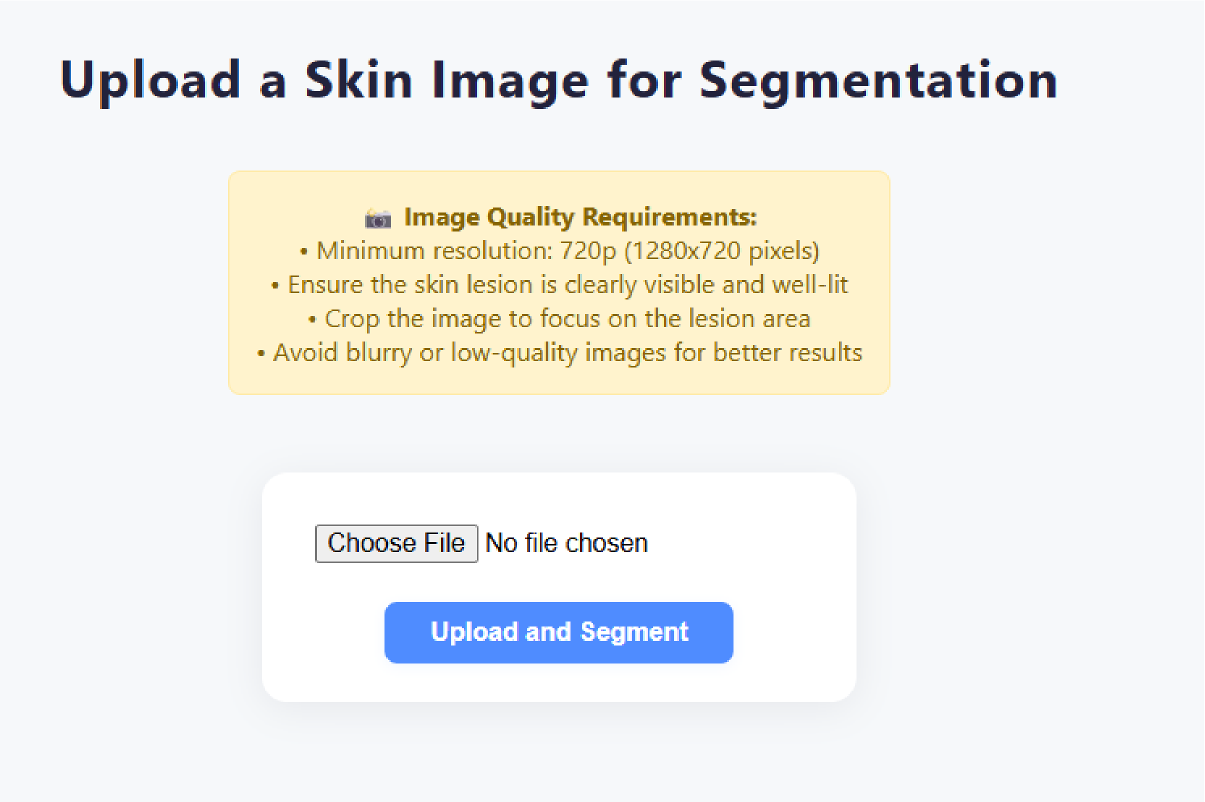

Figure 5.23 Main Page of the Automated System for Segmenting and Classify Skin Lesions Using Deep Learning

Based on the Figure 5.23 shown, it was the main page of the automated system. The main page is simple and understandable for the user to ensure that user will follow the advice on the main page and also avoid user to click the wrong part when upload the image. User only need to upload the skin lesion image and wait for the result.

5.5 Implementation Issues and Challenges

The challenges that had been faced when implement this automated system is the limitation of the dataset had caused the classifier model unable to be high confidence when classify the image exclude from the train and test dataset. For example, when classify the skin lesion is benign or malignant, it might able to classify but some cases it unable to give a high confidence level due to the small dataset on training the classifier model.

Besides, the automated system also unable to classify the unknow object such as the object that are not a skin lesion image. For example, when the user is uploading the picture that are not a skin lesion image, it will also be going to segmentate and classify it, therefore this challenge might cause the loss of the confidence from user to this automated system.

Other than that, is the integration of the models into Flask. For instance, multiple of the large models had increased the memory usage and the start-up time of the system. During the Flask opening process, it might need to wait 1 minute for it to load the Flask because models is large. Other than that, also need to ensure the pre-processing step on Flask are exactly same with the path in training notebook.

Furthermore, is the process on export the segmentation model and classifier model checkpoint and weight are time consuming and required high memory usage. For example, when launching the code of segmentation and classification model, it required like 2 day for export the checkpoint and weight. In the process on launching the code, the laptop might be highly usage in the CPU and it will cause the laptop overheat and become lag. When trying to use GPU to run the pre-trained code will exist error, therefore only able to used CPU to run the pre-trained code.

Chapter 6

System Evaluation and Discussion

6.1 System Testing

6.1.1 Segmentation Model Testing

Table 6.1 Test Cases for Segmentation Model

Test Case	**Test Description**	**Test Data**	**Expected Result**	**Result Show**	**Pass/Fail**
Segmentate skin lesion image (skin lesion area)	Test the segmentation model's ability to segmentate the skin lesion area clearly	Skin lesion image (excluded from the train and test dataset)	Skin lesion area of the input image should be segmentate clearly.	Overlay area had covered on the lesion area.	Pass
Segmentate skin lesion image (low resolution image)	Test the segmentation model's ability on segmentate the skin lesion on low resolution image which is the image lower that 720p.	Skin lesion image that lower than 720p resolution. (excluded from the train and test dataset)	Skin lesion area of the low-resolution image should be segmentate clearly.	Skin lesion area able to segmentate.	Pass

Skin lesion detection (not skin lesion image)	Test the image that not a skin image which is the image that out of distribution.	Random image from the device. (not a skin image)	Should show “This is not skin image” at the segmentation model.	Segmentation model still segmentate the image, although not a skin lesion image.	Fail
Skin lesion detection (darker skin color image)	Test the segmentation model’s ability on segmentate the skin lesion on a darker skin color image.	Darker skin color image. (Image which is darker on the skin color accompany with the skin lesion)	Segmentation model should be able to segmentate the skin lesion area, although the skin is dark in color.	Segmentation model unable segmentate clearly on darker skin.	Fail
Skin lesion detection (image with visible hair)	Test the segmentation model’s ability on segmentate the skin lesion image, if the skin lesion area is covered by some visible hair.	Skin lesion image with visible hair. (Skin lesion image with visible hair and the image is excluded from the train and test dataset)	Segmentation model should be able to segmentate the skin lesion, although the skin lesion is had a bit covered by the visible hair.	Able to segmentate clearly, although the skin lesion area covered by visible hail.	Pass
Skin lesion detection (image with tiny skin	Test the segmentation model’s ability on	Tiny skin lesion area image which also covered	Segmentation model should be able to segmentate	Able to segmentate the tiny skin lesion area.	Pass

lesion area and covered by visible hair)	segmentate the skin lesion, if the skin lesion is is tiny and covered by the visible hair)	by visible hair. (Tiny skin lesion area image and covered by visible hair)	the skin lesion area clearly, although the skin lesion area is tiny and covered by the visible hair.		

6.1.2 Classifier Model Testing

Table 6.2 Test Cases for Classifier Model

Test Case	Test Description	Test Data	Expected Result	Result Show	Pass/Fail
Skin lesion classification (skin lesion type)	Test the classifier model's ability to classify the skin lesion type.	Skin lesion image (excluded from the train and test dataset)	Skin lesion type of the input image should be classified by define its type clearly such as able to classify it is benign or malignant.	Able to classify the skin lesion type.	Pass
Skin lesion classification (High confidence level on malignant skin)	Test the classifier model's ability to classify the malignant skin lesion type in high confidence level.	Malignant skin lesion image (excluded from the train and test dataset)	Malignant skin lesion should be classified in high confidence level.	Able to classify the malignant skin lesion in high confidence level.	Pass
Skin lesion classification (High confidence	Test the classifier model's ability to classify the	Benign skin lesion image (excluded from the train	Benign skin lesion should be classified in high	Unable classify the benign skin in high	Fail

level on benign skin)	malignant skin lesion type in high confidence level.	and test dataset)	confidence level.	confidence level.	
Skin lesion classification (image with visible hair)	Test the classifier model's ability on classify the skin lesion image, if the skin lesion area is covered by some visible hair.	Skin lesion image with visible hair. (Skin lesion image with visible hair and the image is excluded from the train and test dataset)	Classifier model should be able to classify the skin lesion type, although the skin lesion is had a bit covered by the visible hair.	Able to classify the skin lesion image with the visible hair.	Pass
Skin lesion classification (darker skin color image)	Test the classifier model's ability on classify the skin lesion type on a darker skin color image.	Darker skin color image. (Image which is darker on the skin color accompany with the skin lesion)	Classifier model should be able to classify the skin lesion type, although the skin is dark in color.	Unable to classify the skin lesion type on the dark skin color.	Fail
Skin lesion classification (image with tiny skin lesion area and covered	Test the classifier model's ability on classify the skin lesion type, if the	Tiny skin lesion area image which also covered by visible hair. (Tiny skin lesion	Classifier model should be able to classify the skin lesion type, although the	Unable classify the image with tiny skin lesion area and covered	Fail

by visible hair)	skin lesion is tiny and covered by the visible hair)	area image and covered by visible hair)	skin lesion area is tiny and covered by the visible hair.	by visible hair.	

6.1.3 Automated System with Model Implementation Testing

Table 6.3 Test Cases for Automated System with Model Implementation

Test Case	Test Description	Test Data	Expected Result	Show Result	Pass/Fail
Real-time Inference	Test the automated system ability to process the whole workflow from segmentation phrase to classification phrase in speed inference.	Skin lesion image (excluded from the train and test dataset)	The automated system should be showing the result within 1 minute.	Able to show result within 1 minute.	Pass
Invalid file type handling	Test the automated system on handle different file type such as pdf, doc and so on.	PDF file (random PDF file on the device)	The file that not an image will not be show for the user to upload when the user in the upload image phrase.	User unable to upload the file that not an image.	Pass
Oversized file handling	Test the automated system on	Over 10MB file. (random file that over	The oversized file will not be	User unable to upload	Pass

	handle oversized file such as over 10MB.	10MB on the device)	show for the user to upload when the user in the upload image phrase.	oversized file.	
Segmentate skin lesion image on automated system (skin lesion area)	Test the automated system able to remain the segmentation model's ability to segmentate the skin lesion area clearly	Skin lesion image (excluded from the train and test dataset)	Skin lesion area of the input image should be segmentate clearly.	Able to segmentate the skin lesion area clearly.	Pass
Segmentate skin lesion image on automated system (low resolution image)	Test the automated system able to remain the segmentation model's ability on segmentate the skin lesion on low resolution image which is the image lower that 720p.	Skin lesion image that lower than 720p resolution. (excluded from the train and test dataset)	Skin lesion area of the low-resolution image should be segmentate clearly on the automated system.	Able to segmentate the skin lesion area of the low-resolution image clearly.	Pass

Skin lesion detection on automated system (not skin lesion image)	Test the image that not a skin image which is the image that out of distribution on the automated system.	Random image from the device. (not a skin image)	Should show "This is not skin image" at the automated system.	Still segmentate the image that not a skin.	Fail
Skin lesion detection on automated system (image with visible hair)	Test the automated system able to remain the segmentation model's ability on segmentate the skin lesion image, if the skin lesion area is covered by some visible hair.	Skin lesion image with visible hair. (Skin lesion image with visible hair and the image is excluded from the train and test dataset)	Automated system should be able to segmentate the skin lesion, although the skin lesion is had a bit covered by the visible hair.	Able to segmentate the skin lesion, although the skin lesion is had a bit covered by the visible hair.	Pass
Skin lesion detection on automated system (image with tiny skin lesion area and covered by visible hair)	Test the automated system able to remain segmentation model's ability on segmentate the skin	Tiny skin lesion area image which also covered by visible hair. (Tiny skin lesion area image and covered	Automated system should be able to segmentate the skin lesion area clearly, although the	Unable to segmentate the tiny skin lesion area and the system showed "No Lesion	Fail

	lesion, if the skin lesion is tiny and covered by the visible hair)	by visible hair)	skin lesion area is tiny and covered by the visible hair.	Area Found”.	
Skin lesion classification on automated system (skin lesion type)	Test the automated system able to remain the classifier model’s ability to classify the skin lesion type.	Skin lesion image (excluded from the train and test dataset)	Skin lesion type of the input image should be classified by define its type clearly such as able to classify it is benign or malignant.	Able to classify the type of the skin lesion.	Pass
Skin lesion classification on automated system (High confidence level on malignant skin)	Test the automated system able to remain the classifier model’s ability to classify the malignant skin lesion type in high confidence level.	Malignant skin lesion image (excluded from the train and test dataset)	Malignant skin lesion should be classified in high confidence level.	Malignant skin lesion able classified in high confidence level.	Pass
Skin lesion classification on automated	Test the automated system able to	Benign skin lesion image (excluded	Benign skin lesion should be classified	Benign skin unable to classify	Fail

system (High confidence level on benign skin)	remain the classifier model's ability to classify the malignant skin lesion type in high confidence level.	from the train and test dataset)	in high confidence level.	in high confidence level.	
Skin lesion classification on automated system (image with visible hair)	Test the automated system able to remain the classifier model's ability on classify the skin lesion image, if the skin lesion area is covered by some visible hair.	Skin lesion image with visible hair. (Skin lesion image with visible hair and the image is excluded from the train and test dataset)	Automated system should be able to classify the skin lesion type, although the skin lesion is had a bit covered by the visible hair.	Able to classify the skin lesion type, although the skin lesion is had a bit covered by the visible hair.	Pass
Skin lesion classification on automated system (darker skin color image)	Test the automated system able to use classifier model's ability on classify the skin lesion	Darker skin color image. (Image which is darker on the skin color accompany	Automated system should be able to classify the skin lesion type, although the	Unable to classify the skin lesion type when the skin is dark in color.	Fail

	type on a darker skin color image.	with the skin lesion)	skin is dark in color.		
Skin lesion classification on automated system (image with tiny skin lesion area and covered by visible hair)	Test the automated system able to use classifier model's ability on classify the skin lesion type, if the skin lesion is tiny and covered by the visible hair)	Tiny skin lesion area image which also covered by visible hair. (Tiny skin lesion area image and covered by visible hair)	Automated system should be able to classify the skin lesion type, although the skin lesion area is tiny and covered by the visible hair.	Unable to classify the skin lesion type, when the skin lesion area is tiny and covered by the visible hair.	Fail
Summary of result	Test the automated system able to show the original image, overlay image, skin lesion type, malignancy probability and confidence level of the upload data in	Random skin lesion image. (excluded from the train and test dataset)	Automated system should be able to show the original image, overlay image, skin lesion type, malignancy probability and confidence level of the upload data	Able to show the original image, overlay image, skin lesion type, malignancy probability and confidence level of the upload data in the final result	Pass

	the final result.		in the final result.		

6.2 Project Challenges

There have several of challenges when developing this project. The first challenges when developing this this project is to export the checkpoint and the weight of the segmentation and classifier model without losing the accuracy of the model. This challenge was happening because the way to export the model are a bit complicated such as need to know the ways that how the model functioning and how to let the checkpoint and weight of the model store in the path that set to store with it. For example, need to know the backbone of the DeepLabV3 and the structure of the Custom U-Net to simplify the process to export the model without affecting the accuracy of the models.

Furthermore, the second challenges when developing this project is limitation of the dataset. For instance, the training of the segmentation and classifier models are using a small dataset because the legal skin lesion image provided from the website are limited. Although both of the models get a good performance in the result such as able to segmentate and classify the skin lesion image but when processing on some image that are not similar with the dataset that used to train the models the accuracy of the models might drop such as the segmentation model unable to classify the tiny moles and the classifier model hard to get high confidence when classify the benign skin. This incident happens because the ResNet-18 which is the classifier model able to classify the skin lesion by using small dataset for training but the accuracy is not high due to the limited on the skin lesion dataset.

Last but not least, the last challenges when developing this project is implementation of segmentation and classifier model into one automated system. Since, the workflow of the automated system is segmentation model segmentate the image, then pass it to the classifier model. Therefore, the implementation of both models is necessary in this project. The process on implementing of both models are difficult because the segmentation model and classifier model are done by different developer. Hence, to achieved a smooth automated system by combining both models are the most difficult part in this project.

6.3 Objective Evaluation

The first objective of this project is to **developing an automated system for segmenting skin lesion using deep learning**. This objective had been fulfilled as the DeepLabV3 and Custom U-Net model had been successfully implemented into the automated system which both of the segmentation model is achieving high accuracy when segmenting the skin lesion image which both of the models had achieved more than 85% accuracy and able functioning on the automated system. Besides, in the automated system, the both of the segmentation model will be segmentate the image through backend and compare the accuracy of the image, then pass it to the classifier model. This had proved that this objective had been fully fulfilled and contribute a useful function in this project.

The second objective of this project is to **implementing a classifier model into the automated system to classify the moles**. This objective also successfully fulfilled as the ResNet-18 which is the classifier model are successfully implement into the automated system. For example, the classifier model able to classify the skin lesion type. In the process on implement it into the automated system, the confidence level also has been adding into the classifier model to let the user took it as a reference when using the automated system.

The last objective of this project is to **create a user-friendly web interface to show the analyses classification result of the skin lesions**. The last objective had been successfully achieved as the simple web interface had been created in this project. For example, the web interface that accompany with the simple UI had been created in this project to let the user upload the skin lesion image and able to let the user view the summary of result such as original of the image, overlay image (red overlay image), skin lesion type, malignancy probability and confidence level. The simple web interface is giving a direct instruction for the user to upload the image, since the web interface only had one button for the user to click and upload the image. Additionally, the web interface also will only allow the user upload the image file such as jpg, jpeg and so on to avoid the confusion of the automated system and caused the collision of the automated system.

Chapter 7
Conclusion and Recommendation

7.1 Conclusion

In conclusion, the segmentation models accompanied by the classifier model have been successfully implemented into this project, titled **Deep Learning Based Image Segmentation for Dermatological Lesion**, and have been successfully deployed as an automated system for the patient and doctor as a reference tool. This project has successfully fulfilled 3 of the objectives, which are **developing an automated system for segmenting skin lesions using deep learning**, **implementing a classifier model into the automated system to classify the moles**, **and creating a user-friendly web interface to show the analyses classification result of the skin lesions,** as this project had successfully developed an automated system that was able to segment the skin lesion image clearly and classify the skin lesion type as well. Nevertheless, the automated system is also able to show a direct result for the user, such as the original image, the overlay image (red overlay image), the skin lesion type, the malignancy probability and the confidence level for the user as a reference. Additionally, the user-friendly web interface that was created in this project allows users to have an easy interaction with the system, since the system does not have complicated functions and just requires the user to upload the skin lesion image and wait for the result only. However, this project is not flawless; there also exist some flaws in this project. The first flaw of this project is the automated system's unable to get a high confidence level on the benign skin lesion image that was excluded from the training and test dataset. For instance, for the majority of the benign skin lesions of the benign skin type that are not similar to the training and test data, the automated system will get a low confidence level when defining the benign skin type, although it can define it as benign skin correctly but is unable to get a high confidence level. The second flaw of this project is the automated system's inability to segment and classify the tiny skin lesion area when the image is not cropped properly. For example, when the skin is tiny, the user will be required to crop it properly before uploading the image to do a diagnosis. If the user does not crop it properly, then the system will be unable to find the skin lesion image. The last flaw of this project is unable segmentate properly for the skin lesion area that exist on the darker skin colour. For example, when the skin colour of the user is darker than the train and test dataset that used to trained the segmentation model, the automated system will unable to segmentate it clearly.

7.2 Recommendation

There will be several future works that can be done to solve the flaws of the project. The first future work can be done is expand the dataset for the segmentation model. Since, the segmentation models unable to segmentate the moles that exist on the darker skin colour and the tiny skin lesion area. Therefore, by expand the dataset of the skin lesion image which is adding more moles images that exist on the darker skin colour and the tiny skin lesion area will be able to allow the segmentation models able to segmentate the skin lesion area clearly, although the skin lesion area is tiny and exist on the darker skin area.

Furthermore, the future work that can be done to fix the flaw of the automated system will get a low confidence level when defining the benign skin type is implement more trained classifier model into the automated system. For instance, implemented more than one classifier models into the automated system to allow both of the models able to classify the skin lesion image type at the same time and choose the higher accuracy and the high confidence level to show the result for the user which is set a comparison phrase for the classifier model which apply same concept of the segmentation model for the classifier model

Last but not least, if all of these future works able to be done. The automated system will be flawless and able to contribute a useful tool for the community to do skin lesion detection in a costless and convenient ways.

REFERENCES

[1] B. Harangi, "Skin lesion classification with ensembles of deep convolutional neural networks," *Journal of Biomedical Informatics*, vol. 86, pp. 25–32, Oct. 2018, doi: 10.1016/j.jbi.2018.08.006.

[2] J. Y. Wang, E. B. Wang, and S. M. Swetter, "What is melanoma?" *JAMA*, vol. 329, no. 11, p. 948, Mar. 2023, doi: 10.1001/jama.2022.24888.

[3] M. A. Kassem, K. M. Hosny, R. Damaševičius, and M. M. Eltoukhy, "Machine Learning and Deep Learning Methods for skin Lesion Classification and Diagnosis: A Systematic review," *Diagnostics*, vol. 11, no. 8, p. 1390, Jul. 2021, doi: 10.3390/diagnostics11081390.

[4] A. Akram, J. Rashid, M. A. Jaffar, M. Faheem, and R. U. Amin, "Segmentation and classification of skin lesions using hybrid deep learning method in the Internet of Medical Things," *Skin Research and Technology*, vol. 29, no. 11, Nov. 2023, doi: 10.1111/srt.13524.

[5] M. H. Jafari et al., "Skin lesion segmentation in clinical images using deep learning," 2016 23rd International Conference on Pattern Recognition (ICPR), Cancun, Mexico, 2016, pp. 337-342, doi: 10.1109/ICPR.2016.7899656. keywords: {Skin;Lesions;Image segmentation; Malignant tumors;Machine learning; Feature extraction;Lighting;Melanoma;medical image segmentation;skin cancer; convolutional neural network; deep learning},

[6] Wang, Yaoyi & Wu, Qingtao. (2024). Review of Deep Learning Based Segmentation and Recognition of Dermatological Images. International Journal of Computer Science and Information Technology. 3. 32-36. 10.62051/ijcsit.v3n1.05.

[7] S. M. Thwin and H. -S. Park, "Enhanced Skin Lesion Segmentation: DeepLabV3and U-Net with Spatial Attention Mechanisms," 2024 15th International Conference on Information and Communication Technology Convergence (ICTC), Jeju Island, Korea, Republic of, 2024, pp. 1508-1513, doi: 10.1109/ICTC62082.2024.10827768. keywords: {Training;Attention mechanisms;Accuracy;Computational modeling;Skin;Real-time systems;Data models;Lesions;Streams;Skin cancer;DeepLabV3;U-Net;Spatial Attention Mechanism;Skin Lesion Segmentation},

[8] M. Shafiq and Z. Gu, “Deep Residual Learning for Image Recognition: a survey,” Applied Sciences, vol. 12, no. 18, p. 8972, Sep. 2022, doi: 10.3390/app12188972.

[9] Alzubaidi, L., Zhang, J., Humaidi, A. J., Al-Dujaili, A., Duan, Y., Al-Shamma, O., Santamaría, J., Fadhel, M. A., Al-Amidie, M., & Farhan, L. (2021). Review of deep learning: concepts, CNN architectures, challenges, applications, future directions. *Journal of Big Data*, *8*(1). https://doi.org/10.1186/s40537-021-00444-8.

[10] K. Wai Kin, L. Hui Hui, B. Hou Yan, W. Jia Le, “*Image Segmentation of Lesion on Human Skin”* Assignment Code, Universiti Tunku Abdul Rahman, Malaysia 2024.

APPENDIX

POSTER