AN OPTIMAL NEAR-INFRARED SUBCUTANEOUS VEIN EXTRACTION USING DEEP LEARNING

BY

CHAN XIAO JING

A REPORT

SUBMITTED TO

Universiti Tunku Abdul Rahman

in partial fulfillment of the requirements

for the degree of

BACHELOR OF COMPUTER SCIENCE (HONOURS)

Faculty of Information and Communication Technology

(Kampar Campus)

JAN 2022

AN OPTIMAL NEAR-INFRARED SUBCUTANEOUS VEIN EXTRACTION USING DEEP LEARNING

BY

CHAN XIAO JING

A REPORT

SUBMITTED TO

Universiti Tunku Abdul Rahman

in partial fulfillment of the requirements

for the degree of

BACHELOR OF COMPUTER SCIENCE (HONOURS)

Faculty of Information and Communication Technology

(Kampar Campus)

JAN 2022

UNIVERSITI TUNKU ABDUL RAHMAN

REPORT STATUS DECLARATION FORM

Title: AN OPTIMAL NEAR-INFRARED SUBCUTANEOUS VEIN EXTRACTION USING DEEP LEARNING

Academic Session: JAN 2022

I CHAN XIAO JING

(CAPITAL LETTER)

declare that I allow this Final Year Project Report to be kept in Universiti Tunku Abdul Rahman Library subject to the regulations as follows:

1. The dissertation is a property of the Library.
2. The Library is allowed to make copies of this dissertation for academic purposes.

Verified by,

(Author's signature)

(Supervisor's signature)

Address:
No. 220, Kampung Merbau,
32400 Ayer Tawar,
Perak.

Dr. Goh Chuan Meng
Supervisor's name

Date: 20.4.2022

Date: 20.4.2022

FACULTY OF INFORMATION AND COMMUNICATION TECHNOLOGY

UNIVERSITI TUNKU ABDUL RAHMAN

Date: 20.4.2022

SUBMISSION OF FINAL YEAR PROJECT /DISSERTATION/THESIS

It is hereby certified that **CHAN XIAO JING** (ID No: **1804918**) has completed this final year project entitled "**AN OPTIMAL NEAR-INFRARED SUBCUTANEOUS VEIN EXTRACTION USING DEEP LEARNING**" under the supervision of **DR. GOH CHUAN MENG** (Supervisor) from the Department of **COMPUTER SCIENCE**, Faculty of **INFORMATION AND COMMUNICATION TECHNOLOGY**, and **TS DR. TAN HUNG KHOON** (Moderator) from the Department of **COMPUTER SCIENCE**, Faculty of **INFORMATION AND COMMUNICATION TECHNOLOGY**.

I understand that University will upload softcopy of my final year project in pdf format into UTAR Institutional Repository, which may be made accessible to UTAR community and public.

Yours truly,

_____

(CHAN XIAO JING)

DECLARATION OF ORIGINALITY

I declare that this report entitled "**AN OPTIMAL NEAR-INFRARED SUBCUTANEOUS VEIN EXTRACTION USING DEEP LEARNING**" is my own work except as cited in the references. The report has not been accepted for any degree and is not being submitted concurrently in candidature for any degree or other award.

Signature :

Name : Chan Xiao Jing

Date : 20.4.2022

ACKNOWLEDGEMENTS

I would like to express my sincere thanks and appreciation to my supervisor, Dr. Goh Chuan Meng, and my moderator, Ts Dr. Tan Hung Khoon who has given me this bright opportunity to engage in a deep learning research-based project. It is my first step to establish my career in the deep learning field. They have given me a lot of guidance in order to complete this project. When I was facing problems in this project, the advice from them always assists me in overcoming the problems. Again, a million thanks to my supervisor and moderator.

Other than that, to my beloved parents and family, for their patience, unconditional support, and love, and for standing by my side during hard times. I must say thanks to them for their motivation and continuous encouragement throughout the course. Without their support, I will not have enough confidence and courage in striving this project.

ABSTRACT

Intravenous (IV) access is a common and yet important daily clinical procedure that delivers fluids or medication into a patient's vein. However, IV insertion is very challenging where clinicians are suffering in locating the subcutaneous veins due to patients' physiological factors such as hairy forearm and thick dermis fat, and also medical staff's level of fatigue. As a result, the patients are suffering from multiple IV insertions. Despite the fact that numerous studies were being conducted to overcome this limitation, this problem was left unsolved until today.

Therefore, this project has proposed an optimal near-infrared subcutaneous vein extraction technique using deep learning. The proposed model was aimed to be implemented in smart healthcare machines that capture patients' forearms to assist medical staff in locating subcutaneous veins during IV insertion. U-Net, a fully connected network (FCN) architecture was used due to its robustness in biomedical image segmentation. In the project development, the original image was used to train the proposed model without further preprocessing to better generalize to the problem statement and objectives of this project. Besides, data augmentation was applied to increase dataset size and reduce overfitting.

The original U-Net architecture was optimized by replacing upsampling with transpose convolution as well as the additional implementation of batch normalization. The model was furthered trained with different hyperparameters including learning rate, activation function, number of epochs, filter size, and number of layer blocks. After fine-tuning the hyperparameter, unsupervised vein segmentation was implemented by manually selecting 20 checkpoints of true and false vein pixels from the unsupervised forearm images. Then, the saved checkpoints between the forearm images and the predicted output were compared to determine model performance. The proposed model had achieved an accuracy of 0.8871, specificity of 0.9935, sensitivity of 0.7806, and precision of 0.9918, which had successfully fulfilled the defined objectives.

TABLE OF CONTENTS

TITLE PAGE **i**

REPORT STATUS DECLARATION FORM **ii**

FYP THESIS SUBMISSION FORM **iii**

DECLARATION OF ORIGINALITY **iv**

ACKNOWLEDGEMENTS **v**

ABSTRACT **vi**

TABLE OF CONTENTS **vii**

LIST OF FIGURES **x**

LIST OF TABLES **xiii**

LIST OF ABBREVIATIONS **xv**

CHAPTER 1 INTRODUCTION **1**

1.1 Problem Statement and Motivation 1

1.2 Project Objectives 2

1.3 Project Scope 3

1.4 Contributions 3

1.5 Background Information 4

1.5.1 Intravenous (IV) Access 4

1.5.2 Subcutaneous Vein 5

1.5.3 Near-Infrared (NIR) Imaging 6

1.5.4 Deep Learning 6

1.5.5 U-Net Architecture 7

1.5.6 Feature Extraction 8

1.5.7 Image Segmentation 9

1.5.8 Unsupervised Learning 10

1.6 Report Organization 11

CHAPTER 2 LITERATURE REVIEW **12**

2.1 Historical Development of Subcutaneous Vein Feature Extraction 12

2.2 Feature Extraction Based on Deep Learning 14
2.2.1 Feature Extraction using CNN 14
2.2.2 Feature Extraction using DenseNet 16
2.2.3 Feature Extraction using AlexNet 17
2.2.4 Feature Extraction using VGG-Net 18
2.2.5 Feature Extraction using ResNet 19
2.2.6 Feature Extraction using Inception-ResNet 20
2.2.7 Feature Extraction using U-Net 20
2.3 Summary 22
2.4 Proposed Solution 28

CHAPTER 3 SYSTEM MODEL AND DESIGN 29
3.1 Design Specifications 29
3.1.1 Methodologies and General Work Procedures 29
3.1.2 Tools to Use 30
3.1.3 User Requirements 31
3.1.4 System Performance Definition 32
3.2 System Model 35
3.2.1 Initial Optimized U-Net Model Architecture 35
3.2.2 Lightweight U-Net Model Architecture 37
3.3 System Design 38
3.3.1 Data Acquisition 39
3.3.2 Data Augmentation 39
3.3.3 Phase 1 Model Training 42
3.3.4 Phase 2 Model Training 45
3.3.5 Hyperparameter Fine-Tuning 47
3.3.6 Unsupervised Learning 49
3.3.7 Model Prediction on Unsupervised Test Set 52
3.3.8 Hypermodel Performance Evaluation 52
3.4 Implementation Issues and Challenges 53
3.5 Timeline 55

CHAPTER 4 EXPERIMENTS AND SYSTEM EVALUATION 59

4.1 Phase 1 Model Training 59

4.2 Analysis of Model Training on Optimized Network Architecture 60

4.3 Analysis of Model Training on Data Augmentation 63

4.4 Analysis of Model Training on Different Hyperparameters 68

4.4.1 Model Training on Different Learning Rates 68

4.4.2 Model Training on Different Activation Functions 71

4.4.3 Model Training on Different Number of Epochs 73

4.5 Phase 2 Model Training 76

4.6 Hyperparameter Fine-Tuning 77

4.7 Model Evaluation on Unsupervised Test Set Images 80

CHAPTER 5 CONCLUSION AND FUTURE WORK 83

5.1 Conclusion and Objectives Evaluation 83

5.2 Novelties and Contributions 84

5.3 Future Work 84

REFERENCES 86

APPENDIX A

A.1 Weekly Report A-1

A.2 Poster A-14

PLAGIARISM CHECK RESULT B-1

FYP 2 CHECKLIST C-1

LIST OF FIGURES

Figure Number	Title	Page
Figure 1.1	IV access in the arm.	4
Figure 1.2 (a)	Skin and subcutaneous tissue.	5
Figure 1.2 (b)	Illustration of multi-layered skin tissue.	5
Figure 1.3	Example of NIR implementation to localize veins.	6
Figure 1.4	The comparison between classic neural network and deep learning neural network.	7
Figure 1.5	U-Net architecture.	8
Figure 1.6	An example of feature extraction for a cat classification model.	9
Figure 1.7	An example of brain image segmentation.	9
Figure 1.8	An example of unsupervised learning illustration.	10
Figure 2.1	Illustration of feature extraction using deep learning.	14
Figure 3.1	Overview flowchart of the proposed system.	29
Figure 3.2	Initial optimized U-Net model architecture.	35
Figure 3.3	The use of Conv2DTranspose layer to replace UpSampling layer.	36
Figure 3.4	The use of Adam optimizer to replace SGD optimizer.	36
Figure 3.5	The addition of Batch Normalization layers in the contraction blocks.	37
Figure 3.6	Lightweight U-Net model architecture.	37
Figure 3.7	Overview flowchart of the proposed system.	38
Figure 3.8	Subject forearm images and their respective groundtruth images.	39
Figure 3.9	Samples of augmented images after applying the first combination of data augmentation techniques.	41
Figure 3.10	Flowchart of phase 1 model training.	42
Figure 3.11	The performance metrics saved in the history callback.	45
Figure 3.12	Flowchart of phase 2 model training.	45

Figure 3.13	Definition of hyperparameter tuner.	47
Figure 3.14	Definition of hyperparameter search space for learning rate.	47
Figure 3.15	Definition of early stopping.	48
Figure 3.16	Codes to run hyperparameter search.	48
Figure 3.17	Sample images in the unsupervised dataset.	49
Figure 3.18	Flowchart of unsupervised subcutaneous vein segmentation.	50
Figure 3.19	Screenshot of click_event function.	51
Figure 3.20	An example of selecting checkpoints from a forearm image.	51
Figure 3.21	Code to save the true vein checkpoints using pickle library.	51
Figure 3.22	An example of forearm image and its predicted output.	52
Figure 3.23	Gantt chart for the project development timeline (June to September 2021 trimester).	55
Figure 3.24	Figure 3.24 Gantt chart for the project development timeline (January to April 2022 trimester).	56
Figure 4.1	The information generated during the training process.	59
Figure 4.2	Training and validation subset splitting using a ratio of 7:3.	59
Figure 4.3	Example of predicted outputs for the first optimized network architecture.	61
Figure 4.4	Examples of predicted outputs for the second optimized network architecture.	62
Figure 4.5	Comparison of predicted outputs for combination #1 and #2 in terms of noise.	66
Figure 4.6	Comparison of predicted outputs for combination #1 and #2 in terms of broken connection.	66
Figure 4.7	Comparison of predicted output and groundtruth using combination #3.	67
Figure 4.8	Comparison of predicted output and groundtruth using combination #4.	68

Figure 4.9	Comparison of predicted outputs with groundtruth when different epochs were used.	73
Figure 4.10	Dice coefficient curve for the first lightweight model.	76
Figure 4.11	The configuration of learning rate for hyperparameter fine-tuning.	78
Figure 4.12	Result obtained after 17 trials of hyperparameter search.	78
Figure 4.13	The optimum hyperparameters obtained.	78
Figure 4.14	Dice coefficient obtained after retraining the hypermodel using best hyperparameter.	79
Figure 4.15	Dice coefficient curve for the retrained lightweight model.	80

LIST OF TABLES

Table Number	Title	Page
Table 2.1	The comparison of vein feature extraction techniques based on deep learning.	22 – 27
Table 3.1	Hardware used in this project.	30
Table 3.2	Confusion matrix.	32
Table 3.3	Data augmentation techniques used.	40
Table 3.4	Pseudocode of constructing the first optimized architecture.	42
Table 3.5	Phase 1 training process.	44
Table 3.6	Phase 2 training process.	46
Table 3.7	Pseudocode of constructing the optimized lightweight model architecture.	46
Table 3.8	Milestones of the project report writing in current semester and its key deliverables.	57
Table 3.9	Planning for the previous semester and current semester.	58
Table 4.1	Dice coefficient scores for the first optimized network architecture.	61
Table 4.2	Dice coefficient scores for the second optimized network architecture.	62
Table 4.3	Dice coefficient scores and predicted outputs for each data augmentation combination.	64
Table 4.4	Dice coefficient scores and predicted outputs for different learning rates.	69
Table 4.5	Dice coefficient scores and predicted outputs for different activation functions.	71
Table 4.6	Results obtained when different numbers of epochs were used.	74

Table 4.7	Dice coefficient scores for the optimized lightweight model versions.	76
Table 4.8	Summarised selection from training analysis.	77
Table 4.9	Result obtained for model evaluation.	80
Table 4.10	Examples of vein extraction results.	82

LIST OF ABBREVIATIONS

CLAHE	Contrast limited adaptive histogram equalization
CLF	Curve length field
CNN	Convolutional neural network
CPU	Central processing unit
DNN	Deep neural network
EER	Equal error rate
ELU	Exponential linear unit
FCN	Fully convolutional network
GDA	Generalized discriminant analysis
GPU	Graphics processing unit
HD	Hausdorff distance
ISBI	International Symposium on Biomedical Imaging
IV	Intravenous
LBP	Local binary patterns
LDA	Linear discriminant analysis
LEBP	Local extensive binary pattern
LID	Local intensity deviation
NIR	Near-infrared
PCA	Principal component analysis
RAM	Random access memory
ReLU	Rectified linear unit
RNN	Recurrent neural network
ROI	Region of interest
SGD	Stochastic gradient descent
SIFT	Scale-invariant feature transform
SURF	Speeded up robust features
TIF	Tagged image format

Chapter 1 Introduction

1.1 Problem Statement and Motivation

Intravenous (IV) access is one of the fundamental medical procedures that delivers fluids or medication into the vein such as blood withdrawal, drug insertion or transfusion. According to a research conducted by Beecham and Tackling [1], approximately 80 percent of patients would require IV access at some stage during their hospitalization. However, the IV insertion is not a straightforward procedure where clinicians are suffering in locating the subcutaneous vein due to many physiological factors such as hairy forearm, dark skin tone, obese patients, elderly patients, patients in shock, and a high volume of dermis fat [2]. Subsequently, this limitation causes the patients to suffer from venepuncture pain and get injured due to multiple attempts of IV insertion. Besides experiencing increased pain and bruising in the injection area on the arm, Yeguas et al. [3] claimed that multiple IV insertion can lead to delays in medication administration and the increase in the use of medical appliances, injection dose, nurse time, medical cost, and the duration of hospital stay.

Among the methods developed to secure IV access safely, NIR is the optimum imaging technology for subcutaneous vein localization to improve IV success rate. According to Das et al. [4], the deoxygenated haemoglobin in the vein vessels can absorb the infrared light when the light passes through the hand or arm location, or reflects by that location. With the aids of infrared cameras and other related devices, the vein patterns will appear as dark lines. These lines provide a “map” that can be used to guide medical staff for efficient IV insertion [2]. Despite the effectiveness of NIR subcutaneous vein localization in initial stages, later studies reported its limitation in detecting the veins accurately, including first-attempt IV insertion success rates, total procedure time, and lack of availability due to expensive cost [2]. This scenario has made clinical application less common and left the problem unsolved yet till date. Therefore, the use of deep learning techniques is essential and predominant to improve subcutaneous vein localization when using NIR imaging devices and also assist medical staff to carry out IV insertion more accurately.

1.2 Project Objectives

1. **To develop an optimised deep learning architecture for a more precise subcutaneous vein extraction and localization**

   The proposed lightweight U-Net model would improve the extraction accuracy by at least 5% over the existing vein extraction techniques especially for the forearm vein extraction with thick fat layer or body fur. This project has implemented a modified fully convolutional network, U-Net due to its robustness and effectiveness for both vein groundtruth template generation and feature mapping in biomedical image segmentation [5]. The proposed model would also achieve precision of more than 80% to predict more than half of the positive vein pixels accurately.

2. **To assess the model capability on extracting veins from higher-difficulty forearm images**

   The original vein image dataset was boosted by adding forearm images that would bring higher challenges to the extraction process. The newly added data contains more forearm images with thick dermis fur, dark skin tone, and high volume of dermis fat. The purpose was to increase the capability of the proposed model to extract veins from different forearm characteristics as well as to assess model performance when the vein outline was not that visible on a patient's forearm.

3. **To assess the performance of subcutaneous vein segmentation using unsupervised learning technique**

   There was a lack of research studies that implemented unsupervised learning on biomedical image segmentation. Most of the existing studies have used datasets that contain groundtruth, which was the more common and easier way to evaluate model performance. However, obtaining biomedical images for research purpose is expensive, not to mention that generating groundtruth for all training images can be both time-consuming and labour-intensive. Thus, unsupervised learning was implemented in model evaluation phase by evaluating checkpoints chosen from random directions of the veins on the forearm for performance metrics calculation.

1.3 Project Scope

This project has implemented an improved deep learning-based subcutaneous vein extraction technique by proposing a fully connected convolutional network model called U-Net. This optimised lightweight U-Net model was aimed to be implemented in intelligent healthcare machines to replace manual IV insertion. According to Wang and Qin [6], U-Net was a type of modified CNN architecture for fast and precise biomedical image segmentation. As CNN and transfer learning were typically used for biomedical data classification, the U-Net architecture outperformed these deep learning techniques in biomedical image processing, in which the desired biomedical features such as brain tumour and blood vessels can be localized and extracted more efficiently. In this project, the proposed lightweight U-Net model was aimed to localize and extract subcutaneous vein features accurately with the aim to minimize and if possible, eliminate repeated manual IV insertion by medical workers.

Intensive model training was carried out to determine the suitable model architecture optimization, data augmentation combination, hyperparameters, and number of network layers to prevent data overfitting. Due to a small dataset size, the proposed approach has utilized data augmentation to generate a larger batch of image data before feeding the image data into the lightweight U-Net model. The original vein images can be augmented in various transformations including rotation, translation, zooming, and shifting to create a diverse modified training set as well as to expand the size of the training set. This has allowed the model to learn invariance from the diverse deformations efficiently besides resolving the limitations of lacking training data images. Furthermore, unsupervised learning of subcutaneous vein segmentation was explored by comparing checkpoints on forearm images and predicted outputs to determine the number of true positive pixels that can be extracted by the proposed model.

1.4 Contributions

Difficult IV access is common among daily hospitalization procedures. This phenomenon often leads to repeated needle insertion into a patient's vein, which might cause intensified pain and injury due to the increased fragility of venous walls [2]. The proposed optimised lightweight U-Net model can minimize the occurrence of wrong

and repeated IV insertion by assisting medical workers to detect and localize subcutaneous veins for patients whose veins cannot easily be detected.

In addition, medical staff such as doctors and nurses in most countries carry out IV insertion manually regardless of the staff's age, working experience, vision and level of fatigue. Fatigue leads to poor vision with symptoms such as dry or irritated eyes and the retina is difficult to focus. In this fatigue condition, medical staff will be difficult to identify the exact vein location on patients' arms, which causes a higher possibility for the patients to experience pain, injury, or worse, venous thrombosis. Thus, the proposed algorithm was aimed to reduce manual vein localization to be conducted by medical staff.

1.5 Background Information

1.5.1 Intravenous (IV) Access

IV access is one of the most common yet critical clinical procedures for the purpose of patient care and administration, which includes blood sampling, medication injection, parental nutrition injection, transfusion and other blood-based treatments. Conducting IV access is defined as needle insertion or catheter placement into a patient's subcutaneous vein, typically in the arm or dorsal side of hand, and sometimes in the ankle block of foot. In difficult IV access, non-visible and non-palpable veins often results in repeated attempts of IV insertion, causing patients to experience unnecessary pain, anxiety, distress and injury.

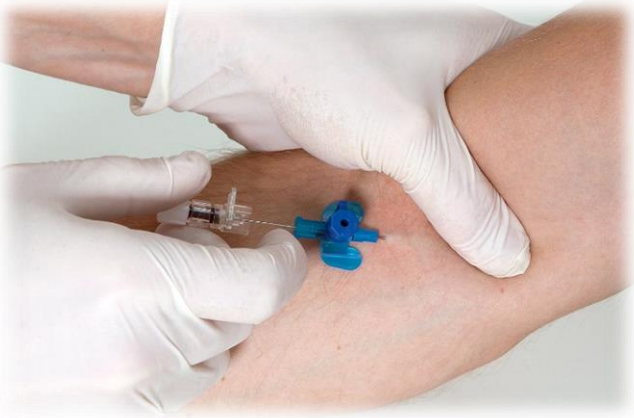


Figure 1.1 IV access in the arm.

Source: Adapted from [7]

1.5.2 Subcutaneous Vein

Subcutaneous veins are the veins located in subcutaneous tissues near the skin surface as shown in Figure 1.5.2 (a) and Figure 1.5.2 (b). To be precise, the veins are located above adipose tissues in the subcutaneous layer. Subcutaneous veins sometimes can be detected by naked eyes, especially during exercise and physical strain. The veins are also visible when medical professionals use tourniquet or elastic band to apply circumferential pressure to the skin and underlying vessels, allowing them to localize a suitable vein for IV insertion.

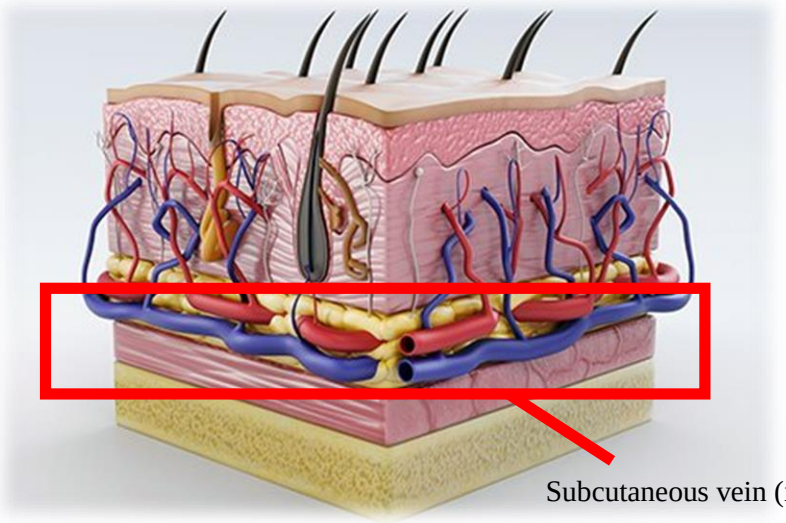


Figure 1.2 (a) Skin and subcutaneous tissue.

Source: Adapted from [8]

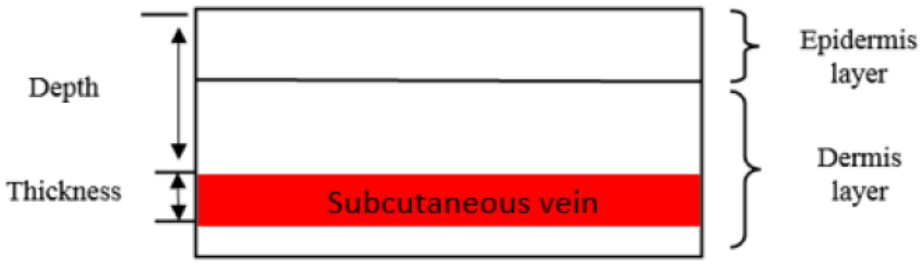


Figure 1.2 (b) Illustration of multi-layered skin tissue.

Source: Adapted from [9]

1.5.3 Near-Infrared (NIR) Imaging

NIR imaging technique is widely used in the biomedical industry and helps in capturing biomedical images for diagnosis, treatment, and research purposes. It utilizes the principle of selective absorption of NIR light waves between 650nm and 1000nm by the deoxygenated haemoglobin in arm, brain, abdomen, and many other tissues to reveal underlying venous network [2]. This imaging technique was implemented clinically by medical devices such as VeinViewer and AccuVein to reveal veins that could not be detected by human eyes or ultrasound. However, unresolved issues regarding vein localization and difficult IV access still exist despite the use of such technology.

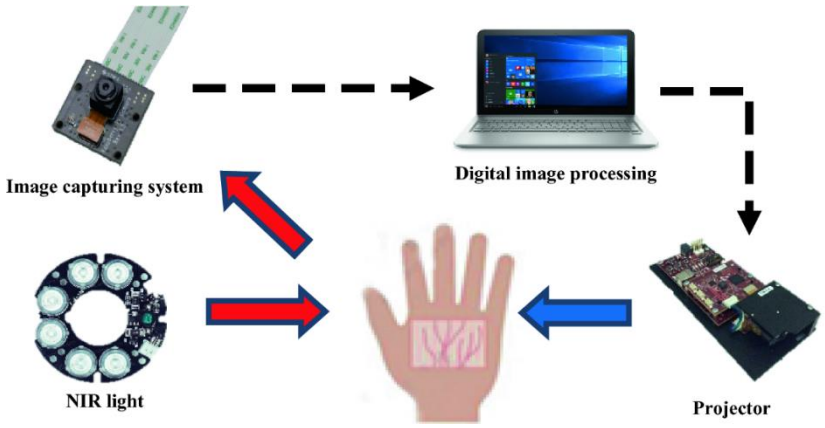


Figure 1.3 Example of NIR implementation to localize veins.

Source: Adapted from [10]

1.5.4 Deep Learning

Deep learning is a subset of the machine learning family that can be implemented using artificial neural networks. Artificial neural network is inspired by the workings of numerous neurons in the human brain that includes automatic signal transmission, data processing, and decision making. Artificial neural network infrastructure possessed by deep learning depicts interconnecting nodes called neurons in the network layers that can learn features by themselves from different types of data automatically without the need of manual coding by programmers.

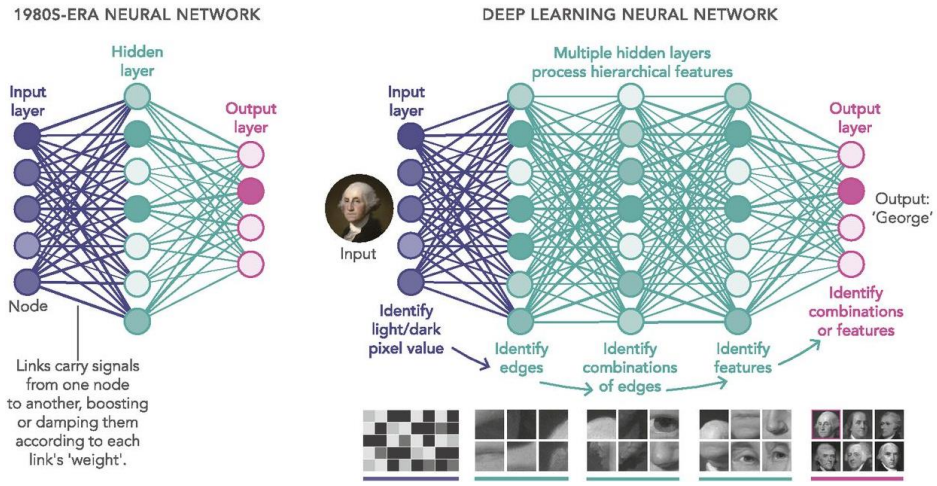


Figure 1.4 The comparison between classic neural network and deep learning neural network. Source: Adapted from [11]

1.5.5 U-Net Architecture

U-Net is a modified CNN architecture favoured by the biomedical image processing field due to its robustness and great performance in biomedical image segmentation. This end-to-end fully convolutional network can work with very few training images and yet guarantee accurate segmentation results. Unlike CNN architecture, U-Net consists of encoder layers which reduce the dimensions of the image, and decoder layers that repair the detailed features in the spatial dimension of the image during up-sampling [5].

The network architecture consists of a contracting path, the bottleneck, and an expansion path. The contracting path contains unpadded contraction convolutional layers, with each convolution followed by ReLU activation and max-pooling layers for downsampling. To enable the model to learn the complex features in the images effectively, the feature maps after each block of layers are doubled [5]. The bottleneck layer mediates in the middle of the contracting path and the expansion path with the use of normal convolutional layers followed by an up-convolutional layer.

Similar to the contracting path, the expansion path also has multiple blocks of layers. In each layer block, the features are being inputted into convolutional layers followed by an upsampling layer. The feature map in the expansion path gradually reduces itself by half to maintain the symmetry of upsampling [5]. The merge layer for

each layer block then concatenates both the previous convolutional layer and the upsampling layer so that the features learned can be used by the current convolutional layer to reconstruct the image features effectively. The final layer is a 1×1 convolution which applies sigmoid function, in which the value of each image pixel is the probability of being the vein feature [6].

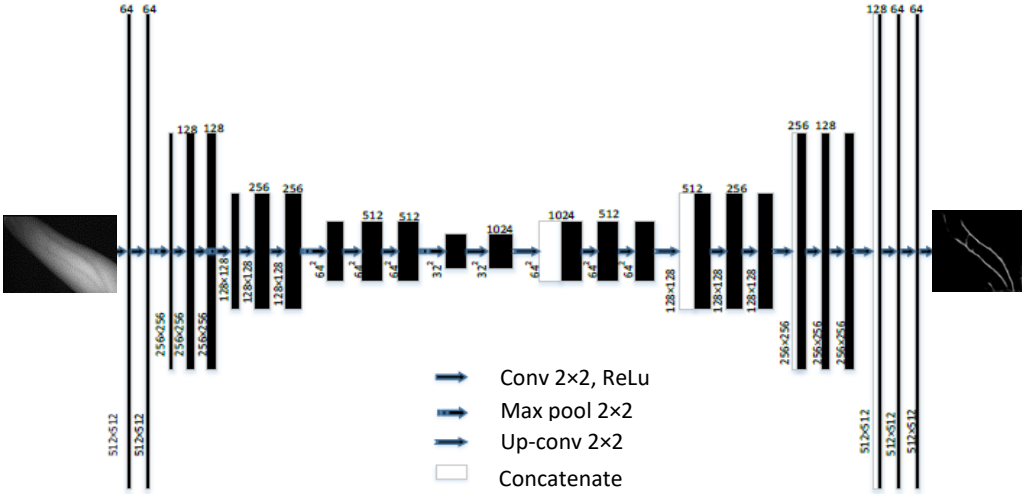


Figure 1.5 U-Net architecture.

Source: Adapted from [6]

1.5.6 Feature Extraction

Feature extraction is a process of finding the most useful, informative and non-redundant set of features to improve the performance of data processing and model training [12]. The initial input data usually comes with a large amount of redundant features. Thus, it is necessary to perform dimensionality reduction to select and/or combine relevant features as well as to remove those non-dominant features [13]. Besides reducing training time and model complexity, the extracted features are still capable of describing the original input data precisely and wholly without losing important information about the data [13].

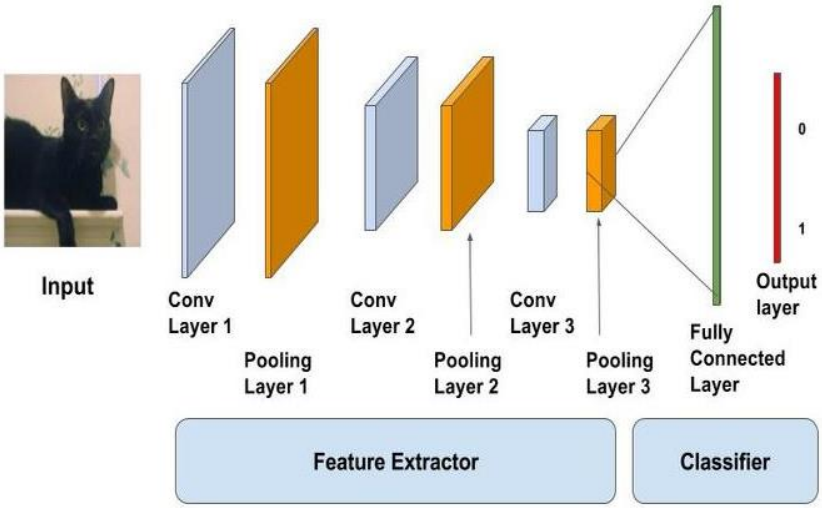


Figure 1.6 An example of feature extraction for a cat classification model.

Source: Adapted from [14]

1.5.7 Image Segmentation

Image segmentation is being commonly used in image processing to segregate an image into multiple segments or regions of pixels [15]. Each region is then converted into a mask with each pixel of the image being given a label, indicating whether the pixel belongs to the class. Image segmentation becomes increasingly important in a lot of medical image research and clinical applications [16]. For example, this technique is used to detect and label pixels in a 2-dimension image that represents a network of blood vessels in a patient's forearm or fundus, or a 3-dimension voxel that indicates a tumour in a patient's breast or brain.

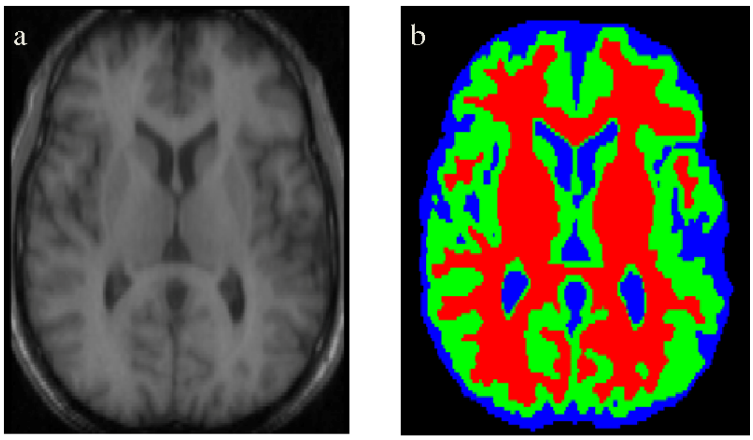


Figure 1.7 An example of brain image segmentation.

Source: Adapted from [17]

1.5.8 Unsupervised Learning

Unsupervised learning is a technique that uses machine learning algorithms to analyze, train, and infer unlabelled datasets, or data without groundtruth. Unlike supervised learning, unsupervised learning cannot be directly implemented in common machine learning problems such as classification and regression because the model does not have data pairs to train on. Instead, it is used to discover the underlying pattern of the data based on similarities and differences. Although unsupervised learning is more computationally complex, it can be implemented in biomedical image segmentation by clustering the image pixels into their respective groundtruth classes without having the groundtruth being annotated.

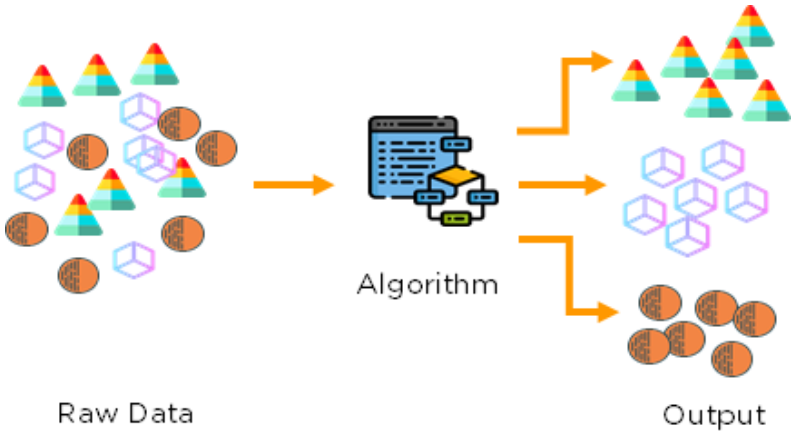


Figure 1.8 An example of unsupervised learning illustration.

Source: Adapted from [18]

1.6 Report Organization

Firstly, chapter 1 of this project, as elaborated in the previous sections, gave a detailed introduction that has included problem statements, motivations, project objectives, scope, and contributions. Relevant background information and technical terms were provided to give more understanding about the foundation of this project.

Secondly, literature review of previous studies that were relevant to subcutaneous vein extraction was elaborated in chapter 2. This section gave a historical development of vein extraction on traditional approaches as well as detailed review on subcutaneous vein extraction using different deep learning techniques. After that, the proposed solution was discussed briefly based on the information obtained from the reviewed studies.

Thirdly, chapter 3 was about the detailed explanation of the proposed system model and design. This chapter has first included design specifications that consisting of methodologies and general work procedures, tools to use, user requirements, and system performance definition. Then, system model subchapter has described about the initial optimized U-Net model architecture and the lightweight U-Net model architecture. Subsequently, the system design was divided into several stages, which were data acquisition, data augmentation, phase 1 model training, phase 2 model training, hyperparameter fine-tuning, groundtruth prediction on unsupervised test set, and hypermodel performance evaluation. Implementation issues and challenges as well as project timeline were included in this chapter.

Next, chapter 4 discussed about the experiments conducted and system evaluation. The results achieved in each stage, which were phase 1 model training, phase 2 model training, hyperparameter fine-tuning, and unsupervised prediction, were being analysed in a detailed manner.

Lastly, the conclusion and future work of the project were elaborated in chapter 5. This chapter has included objectives evaluation as well as the novelties and contributions delivered throughout the entire project development. Future work was discussed to provide recommendations for further improvements on the proposed model.

Chapter 2 Literature Review

2.1 Historical Development of Subcutaneous Vein Feature Extraction

Before deep learning became the center of attention for feature extraction tasks, vein extraction and recognition studies were conducted using traditional or non-training-based feature extraction techniques, such as:

- feature extraction based on vein patterns,
- feature extraction based on dimensionality reduction,
- feature extraction based on local binary patterns, and
- feature extraction based on image transformations.

Firstly, vein pattern extraction was a technique that used preprocessed vein images as inputs to the feature extraction algorithms to extract vein topology or curvature information. Some examples of vein pattern-based feature extraction algorithms were Gabor filter, Hausdorff distance (HD), curve analysis, and local descriptors such as local extensive binary pattern (LEBP) and local intensity deviation (LID).

This technique was heavily dependent on the preprocessing procedure, in which the performance will be better if the vein visibility in the images is higher. However, these algorithms needed to be manually calibrated and would highly affect the performance if their values were not set appropriately. Besides, the classification for vein pattern-based extracted features was fast for small training sets, but time-consuming for big databases.

Secondly, dimensionality reduction was one of the data transformation techniques to transform data from high dimensions to low dimensions. It aimed to make the transformed data more meaningful compared to the original data as well as to reduce data sparsity in high dimensional space. Some examples of the techniques were PCA, linear discriminant analysis (LDA), and repeated line tracking.

This technique was more systematic to decrease the feature vector dimensionality and remove noise information. However, there was a high risk that the images that had undergone dimensionality reduction had lost too much information on the desired feature [19]. It might also be computationally expensive because some

dimensionality reduction algorithms will build relation graphs in the background when finding relationships between subjects [20].

Thirdly, local binary patterns (LBP) was a simple yet efficient texture and visual descriptor that can label pixels near a central value using a threshold. Each binarized pixel will be concatenated and converted into a decimal value that denotes a pixel from the original images. As a result, a new LBP image, in which it contains better representations of the original image, will be extracted.

Feature extraction using LBP was able to improve recognition time and was more robust against local region recognition. The derived learning feature can minimize quantization error and make the binary code more discriminative [21]. However, parameter values were not learned from the system and need to be manually defined. Any changes to the values will highly affect the accuracy [21].

The fourth traditional feature extraction technique was based on image transformations. Transformation methods such as spatial curve filters, radon transform, or discrete wavelet transform will be applied on the extracted ROIs for further prediction using a classifier-based model. Some examples of the techniques were Radon transform [22], curve length field (CLF) estimation [23], Hilbert-Huang transform, and Dual tree complex wavelet transform [24].

Previous studies that implemented this feature extraction technique have mostly obtained low EERs of approximately 0.20%, but Yang et al. [23] claimed that the filter generated might be too well-fitted on the vein features. The image transformation functions need manual parameter calibration. The feature extraction process can be sensitive to parameters such as scaling factors and rotational degrees if the values were not set appropriately.

Overall, the major drawback of traditional feature extraction approaches was that the important features in the images need to be chosen and defined manually by the researchers rather than learning the features automatically by the model. This process could be complicated and inefficient especially when there was a large number of classes needed to be classified as well as a large number of parameters needed to be fine-tuned [25].

The pitfall of handcrafted feature extraction algorithms can be improved greatly by implementing deep learning approaches in biomedical image processing and feature extraction. Important features can be learned automatically by the network model without manual feature selection [26]. Different deep learning techniques used in vein feature extraction were discussed in the next section.

2.2 Feature Extraction Based on Deep Learning

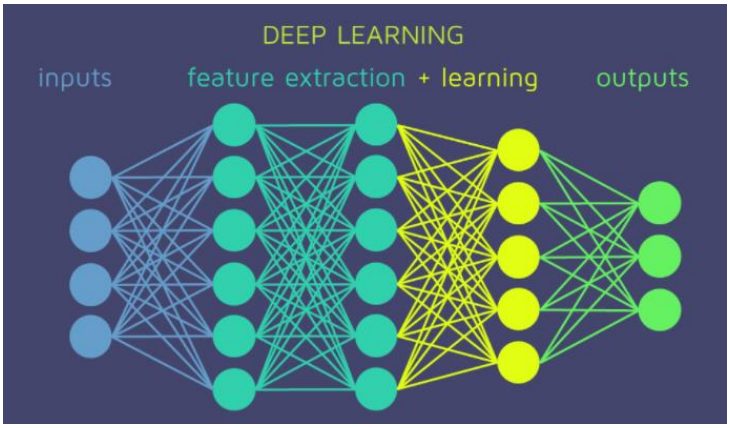


Figure 2.1 Illustration of feature extraction using deep learning.
Source: Adapted from [27]

The automation and optimization behaviour of deep learning framework has made feature extraction to be conducted in a more efficient way. Deep learning models such as CNN and U-Net consist of feature extraction layers that are able to extract optimised features from the input images automatically. The neurons in the hidden layers will learn the useful features from the images before reaching the output layer without involving handcrafted feature descriptors.

2.2.1 Feature Extraction using CNN

Qin and El-Yacoubi [28] developed a CNN to learn robust features and attributes from raw pixels automatically for finger vein representation. The CNN consisted of two convolutional layers with max-pooling and normalization, followed by a fully connected softmax layer to predict the probabilities of pixels. This model was incorporated with a FCN to recover missing vein features during the extraction process.

This approach was able to learn robust features from a large-scaled dataset. However, the automatic vein pixel labelling scheme based on traditional baseline algorithms failed in segmenting images with poor illumination, leading to poor pixel classification.

Qin and El-Yacoubi [29] further proposed a patch-based DNN system for finger vein verification that can label high and low quality images automatically. Two DNN models were constructed with the purposes of splitting the original vein images into rectangular patches and assessing vein image quality respectively. The image patches were divided in an overlapping manner to conserve the connectivity of vein patterns [29]. This system eliminated the use of hand-craft descriptors with manually selected attributes and reduced unnecessary preprocessing work. However, their approach still failed to classify a number of low and high quality vein images after fine-tuning the parameters.

Fang et al. [30] proposed two lightweight CNN models, which was two-channel network and two-stream network for finger vein extraction. Depending on these two networks, a combined selective network was built to automatically adopt the most suitable network for different subjects based on the intra-class differences of their vein features. This proposed network explored different types of mini-ROI extraction methods to remove the impact of image displacement. However, minimum preprocessing was done on the ROI images. There might still be a large displacement in the input images that can inhibit the ability of the network to learn better diverse features.

Das et al. [31] proposed a finger vein identification system by constructing a CNN model that contained five convolutional layers along with three max-pooling layers, a ReLU layer, and a fully connected softmax loss layer. Fixed dimension input images were used to eliminate severe resizing operations. To facilitate the feature extraction process, rotation and vertical translation of the images were compensated by normalizing the images before further enhancing the images with CLAHE. This system was able to extract vein features from both high and low quality images, but the use of low learning rate can cause data overfitting, leading to inaccurate performance.

Boucherit et al. [32] proposed a finger vein identification system using a deeply-fused convolutional network. While most of the existing topologies focused on using a single CNN for image processing, segmentation and feature extraction, they enhanced

the current topologies by using Merge CNN, which used several CNNs with short paths. This architecture was novel and proved to be able to increase the finger vein recognition rate compared to normal CNN [32]. However, training multiple similar networks can be computationally expensive and time consuming.

Cherrat et al. [33] proposed a CNN as an automatic feature extractor for their vein identification system combined with softmax and Random Forest model for vein classification. The CNN model contained five hidden layers with the use of ReLU activations and max-pooling layers in each convolutional block. Softmax function was applied at the output layer to predict extracted vein features. This model had implemented data augmentation to reduce data overfitting and increase the size of the training dataset. However, the model was relatively small and its ability to extract accurate vein features was questionable.

Kuzu et al. [34] proposed an advanced CNN named Vein-CNN which was specifically designed for finger vein recognition. The model contained six convolutional layers, four max-pooling layers, two fully connected layers, and an output softmax layer. The convolutional layers were divided into four blocks with each block followed by batch normalization, ReLU, and down-sampling layers with the use of different kernel sizes in each block. Despite its simplicity, the model was able to outperform other advanced architectures. However, the dataset used was quite small and there was no implementation of data augmentation to increase data diversity.

2.2.2 Feature Extraction using DenseNet

Song et al. [35] also proposed a DenseNet-161 architecture to extract vein features from a composite image that combined two finger vein images to minimize noise susceptibility of vein pixels. Since DenseNet was more difficult to train due to the increase of hidden layers, bottleneck layers consisted of BatchNormalization layers, ReLU layers, convolutional layers and transition layers were added between dense block layers to prevent the growth of feature map size. This network was able to achieve higher accuracy than normal CNN, but the training time was long due to large numbers of hidden layers and transition layers.

Kuzu et al. [36] proposed a modified DenseNet-161 architecture by adding a custom embedded layer before the classification layer to extract vein features. The network also applied transfer learning by employing the weights of pre-trained Imagenet models to initialize all the layers except the custom embedder layer and was trained using Siamese strategy. The implementation of pre-trained network parameters was proved to be more stable and guaranteed lower EERs. However, the training of two networks simultaneously was computationally expensive and might slow down the training process.

Similarly, Noh et al. [37] proposed a DenseNet-161 model for finger vein recognition. Unlike the previous two DenseNet-161 studies, this approach adopted score-level fusion of two CNNs to fuse both vein texture images and shape images. Thus, both the ample information as well as compact information of vein images were used to train the DenseNet-161 network. Noh et al. [37] had set the model growth rate as 48 to change the channel size of input feature map in each block. This approach had improved recognition performance by considering two input image types, but processing time was long with the use of heavy models and large input size.

2.2.3 Feature Extraction using AlexNet

A finger vein extraction system was proposed by Liu et al. [38] that adopted a seven-layer AlexNet consisting of 5 convolutional layers and 2 fully-connected layers. All the layers used the same parameter values: kernel size of 5×5 and pooling size of 2×2. Less time was needed to train the network, but the preprocessing steps were ineffective to generate clearer input images for vein feature extraction.

Meng et al. [39] proposed a relatively simple and easy-to-implement pretrained AlexNet for finger vein extraction and recognition. The feature extraction pipeline did not include complex image preprocessing and directly inputted the images into the fine-tuned model to extract representative feature vectors. This approach implemented data augmentation to increase dataset size and prevent overfitting. However, network parameters were not fine-tuned to suit the extraction domain. The model was trained in CPU memory which slowed down the whole development progress.

Raghavendra et al. [40] had applied transfer learning of deep CNN using pretrained AlexNet architecture to extract vein features from video frames. The existing model architecture was augmented by adding additional seven layers to improve detection reliability and reduce overfitting issues, which led to low error rates. However, it might add complexity to model training since the network had more layers.

Besides, Fairuz et al. [41] implemented transfer learning of AlexNet to build a CNN model in their proposed finger vein identification system. The use of a pre-trained network allowed parameter adjustment to adapt the model to other tasks easily. The implementation of a pre-trained network was less complex, time-saving and computationally cheaper to build. However, the fully connected layers of the pre-trained network were not fine-tuned, in which the model might be difficult to adapt to a new domain.

2.2.4 Feature Extraction using VGG-Net

Hong et al. [42] proposed a pre-trained VGG-16 for finger vein recognition. This model was trained and fine-tuned on 224×224 difference images obtained from input images and enrolled ROI images. To tackle the overfitting problem, data augmentation and dropout layers had been applied to increase dataset size and to reduce connections between fully connected layers. However, the model worked poorly on low-quality images as it was observed that EER of low-quality images was very high compared to good- and mid-quality images.

Al-Johania and Elrefaei [43] proposed transfer learning using AlexNet, VGG-16 and VGG-19 for hand vein feature extraction and classification. The VGG-19 model is larger than VGG-16 where VGG-19 contains four convolutional layers in the fourth and fifth convolutional blocks while VGG-16 contains three convolutional layers in those blocks. Vein images were inputted into these models without preprocessing and features were extracted by the modified FC layers of the models. Both VGG-16 and VGG-19 outperformed AlexNet in overall, indicating that large networks with adequate augmented data were workable for vein extraction. However, training on too many epochs can lead to data overfitting, resulting in over-optimistic accuracies.

Harshan et al. [44] also implemented the VGG-16 model for their finger vein recognition system. Similar to the system proposed in [42], data augmentation was applied to perform changes in padding, dimension cropping and flipping of images. Instead of inputting the images directly into the model, feature extraction was first conducted using the LBP algorithm. Then, the extracted features were used as input to the model. This double feature extraction pipeline had improved the recognition rate and EER. However, transformation techniques were inadequate to generate a large diversity of vein images.

2.2.5 Feature Extraction using ResNet

Mohaghegh and Payne [45] had investigated dorsal hand vein identification using three three pre-trained CNNs, which were AlexNet, ResNet50 and ResNet152. After fine-tuning the output layer to better suit the output domain, augmented vein images were inputted into the models. The results obtained showed that ResNet models achieved better recognition performance than AlexNet due to deeper network architecture to learn complex features. However, since there was only 0.02% difference between testing accuracies of ResNet50 and ResNet152, this indicated that ResNet50 was adequate for vein feature extraction tasks besides reducing training time on a smaller network.

Tang et al. [46] proposed a vein localization system by using ResNet34 as encoder. The decoder of the model contained feature extractor, a dilated convolution module, as well as a transposed convolution module to recover vein feature images. ResNet34 was chosen as the feature extractor due to the skip connection characteristic that can skip a few hidden layers and connect directly to the desired layer. Thus, regularization would not perform on those layers to prevent performance degradation. Although this integrated model outperformed other architectures such as ResNet18 and ResNet50, the accuracy was still low compared to other reviewed studies, indicating that integrating three architectures might not be a good choice for vein extraction.

A deep neural network consisting of VGG-19 and ResNet was proposed by Tao et al. [47] for bidirectional feature extraction to improve finger vein recognition rate. Instead of training one-way from top to bottom or from left to right, the proposed method applied two feature extraction processes in both directions by applying feature

concatenations to obtain more features. Tao et al. [47] claimed that the combination of VGG-19 that had non-branching structure and ResNet with residual connections was the most superb choice for bidirectional feature extraction which proved by the high accuracies obtained in their experiments. However, this approach resulted in the longest training time compared to other deep learning based feature extraction studies.

2.2.6 Feature Extraction using Inception-ResNet

Zhang et al. [48] proposed a modified Inception_ResNet_v1 network to increase network depth for effective vein feature extraction. Initially, a seven-layer network was constructed that consisted of six convolution layers and one max-pooling layer. Then, three Inception-ResNet blocks were used to speed up network convergence by implementing residual connections and two Reduction modules were used to reduce model complexity when extracting representative vein features [48]. This network was comprehensive and novel to be implemented on large datasets, but this may incur long training time and complexity due to the addition of many layers.

2.2.7 Feature Extraction using U-Net

Jalilian and Uhl [49] developed a finger vein recognition with the comparison between three semantic deep learning models, which were U-Net, RefineNet and SegNet. U-Net was trained on an SGD optimizer with learning rate of 0.08 for 300 epochs. Similarly, SegNet was trained SGD optimizer with a smaller learning rate of 0.003 for 30,000 epochs. RefineNet was trained on Adam optimizer with a much smaller learning rate of 0.0001 for 40,000 epochs. Each network used different sets of training parameters which might lead to biased results, but it was notable that U-Net was able to obtain the lowest EER in overall and outperformed maximum curvature extraction technique [49].

Marattukalam and Abdulla [5] proposed a modified U-Net architecture that used a Gabor filter in the first encoding block. The ROIs were first thresholded to a foreground value to label vein and background pixels. Then, morphological operations which were dilation and erosion were applied to remove noises from the images. Skeletonized images were obtained and trained on the model by using Adam optimizer

and learning rate of 0.00015. This technique was able to extract palm vein features with good accuracy. However, dice coefficient was low with approximately 70% which made the generated groundtruth images less reliable.

Moreover, Wang and Qin [6] implemented an end-to-end U-Net model for vein feature extraction. U-Net contains an encoding path that gradually reduces spatial dimensions of inputs for extraction purpose and a decoding path that gradually recovers spatial dimensions of the images. The vein images were first segmented using four baseline algorithms and binarized by a threshold of 0.5 to label the vein pixels. The binarized images were used for model training to get the probability score of each pixel and to generate predicted output through thresholding. The extraction process was fast and simple with a moderate network size yet ensured satisfied outputs. However, vein pixel labelling was relied on multiple handcrafted algorithms which might not be effective in the long term.

Zeng et al. [50] proposed a deformable convolutional network by modifying the standard U-Net architecture that can capture the complex venous structural features effectively. The convolutional layers in standard U-Net were replaced by deformable layers to adjust receptive fields adaptively and residual recurrent layers for effective depth mining and feature accumulation [50]. This modified architecture was able to obtain a low EER and high extraction ability compared to the original U-Net. However, the integration of U-Net, RNN, ResNet and deformable neural networks might be complicated and heavy duty to be trained.

2.3 Summary

Table 2.1: The comparison of vein feature extraction techniques based on deep learning.

Author (year)	Technique used	Pros	Cons
(A) Vein Feature Extraction using CNN			
Qin and El-Yacoubi [28]	Standard CNN incorporated FCN to recover missing vein features during the extraction process.	Able to learn robust features from a large-scaled dataset.	Automatic vein pixel labelling scheme failed to segment images with poor illumination, leading to poor pixel classification.
Qin and El-Yacoubi [29]	A patch-based DNN consisted of two DNN models.	Eliminated the use of handcraft descriptors and unnecessary preprocessing work.	Still failed to classify a number of low and high quality vein images after fine-tuning the parameters.
Fang et al. [30]	A combined selective network that consisted of two-channel and two-stream CNNs for vein feature extraction.	Implemented different ROI extraction methods to minimize image displacement. Required shorter training time.	Minimum data preprocessing was done. Training and testing stages were conducted in a disjoint manner. Output results might be biased.
Das et al. [31]	A CNN with 5 convolutional layers, 3 max pooling layers, a ReLU layer, and a softmax layer.	Able to extract features from both high and low quality images. Used fixed image size to reduce effort for resizing operation.	Used low learning rate which might cause slow training progress, difficult to converge, and data overfitting.

Author (year)	Technique used	Pros	Cons
Boucherit et al. [32]	Merge CNN with late fusion by training four vein datasets with different qualities using the same CNN topology.	Increased vein recognition rate. Able to train multiple CNN models in parallel.	Small training set compared to test set, which might lead to data overfitting. Computationally expensive and time consuming.
Cherrat et al. [33]	CNN as automatic feature extractor combined with softmax and Random Forest model.	Implemented data augmentation to reduce data overfitting and increase the size of the training dataset.	Small model was built, leading to questionable feature extraction outcomes.
Kuzu et al. [34]	Advanced Vein-CNN specifically designed for finger vein recognition.	Simple yet outperformed other advanced architectures.	Small dataset was used. Did not implemented data augmentation to increase data diversity
(B) Vein Feature Extraction using DenseNet			
Song et al. [35]	A DenseNet-161 model by adding BatchNormalization and transition layers.	Achieved higher accuracy than normal CNN.	Long training time due to large numbers of hidden and transition layers.
Kuzu et al. [36]	A modified DenseNet-161 model that applied transfer learning of pretrained weights.	Pretrained parameters were more stable and effective to extract features. Able to train on three different kinds of veins.	Computationally expensive and time-consuming to train two networks in parallel.

Author (year)	Technique used	Pros	Cons
Noh et al. [37]	A DenseNet-161 model that adopted score-level fusion of two CNNs.	Improved recognition performance by considering two types of input image.	Long processing time with the use of heavy models and large input size.
(C) Vein Feature Extraction using AlexNet			
Liu et al. [38]	A pretrained AlexNet model.	Required less time to train the pretrained AlexNet model. Some parameters were adjusted to suit the problem domain of CNN.	Less effective in the image preprocessing and parameter adjusting phases, which might affect the vein feature extraction process.
Meng et al. [39]	A simple and easy-to-implement pretrained AlexNet model.	Implemented data augmentation to increase dataset size and prevent overfitting.	Network parameters were not fine-tuned to suit the extraction domain. Slow training progress in CPU memory.
Raghavendra et al. [40]	Transfer learning of deep CNN using pretrained AlexNet with the addition of seven layers.	Modified network was able to obtain low error rates. New layers had been fine-tuned to maintain the adaptation and robustness of the original framework.	Increase complexity and time to train the network. Minimum data preprocessing was done, resulted in ROIs with low quality.

Author (year)	Technique used	Pros	Cons
Fairuz et al. [41]	The transfer learning of a pretrained AlexNet to build a CNN and test the accuracy of the model from three experiments.	Less complex and computationally cheaper. Save time because no need to build the whole model from scratch. Used an in-house database instead of a public database.	Did not work on fine-tuning the pretrained AlexNet model. This might be difficult for data to adapt to the new model domain.
(D) Vein Feature Extraction using VGG-Net			
Hong et al. [42]	A pretrained VGG-16 model with the implementation of data augmentation.	Applied data augmentation and dropout layers to prevent overfitting issues.	Worked poorly to extract vein features from low-quality images, resulting in extremely high EER.
Al-Johania and Elrefaei [43]	Transfer learning using AlexNet, VGG-16 and VGG-19.	Both VGG-16 and VGG-19 had outperformed AlexNet.	Training on too many epochs resulted in over optimistic accuracies due to data overfitting.
Harshan et al. [44]	A VGG-16 model that implemented feature extraction using LBP algorithm before training the model.	The double feature extraction pipeline had improved the recognition rate and EER.	Transformation techniques were inadequate to generate a large diversity of vein images.

Author (year)	Technique used	Pros	Cons
(E) Vein Feature Extraction using ResNet			
Mohaghegh and Payne [45]	Dorsal hand vein identification using AlexNet, ResNet50 and ResNet152.	ResNet models achieved better recognition performance than AlexNet due to deeper network architecture to learn complex features.	Small difference between testing accuracies of ResNet50 and ResNet152, indicated that smaller network was adequate for feature extraction.
Tang et al. [46]	A model that used ResNet34 as the encoder, and feature extractor, dilated and transposed convolution modules as the decoder.	This integrated model outperformed other architectures such as ResNet18 and ResNet50.	The accuracy was still low, indicating that integrating three architectures might not be a good choice for vein extraction.
Tao et al. [47]	A deep neural network consisted of VGG-19 and ResNet for bidirectional feature extraction.	Novel deep learning architecture. Obtained high accuracy.	Long training time.
(F) Vein Feature Extraction using Inception-ResNet			
Zhang et al. [48]	A modified Inception_ResNet_v1 network with the addition of Inception-ResNet blocks and Reduction modules.	Comprehensive and novel to be implemented on large datasets.	Might incur long training time and complexity due to the addition of many layers.

Author (year)	Technique used	Pros	Cons
(G) Vein Feature Extraction using U-Net			
Jalilian and Uhl [49]	A finger vein recognition with the comparison between U-Net, RefineNet and SegNet.	U-Net was able to obtain the lowest EER overall.	Each network used different sets of training parameters which might lead to biased results.
Marattukalam and Abdulla [5]	A modified U-Net architecture that used a Gabor filter in the first encoding block.	Able to extract palm vein features with a good accuracy.	Dice coefficient was low, which made the generated groundtruth images less reliable.
Wang and Qin [6]	An end-to-end U-Net model for vein feature extraction.	The extraction process was fast and simple with a moderate network size yet ensured satisfied outputs.	Vein pixel labelling was relied on multiple handcrafted algorithms which may not be effective in the long term.
Zeng et al. [50]	A deformable convolutional network by modifying the standard U-Net architecture.	Able to obtain a low EER and high extraction ability compared to the original U-Net.	The integration of U-Net, RNN, ResNet and deformable neural network might be complicated and heavy duty to be trained.

2.4 Proposed Solution

It was observed that feature extraction using deep learning approaches were more robust to learn features from the biomedical images in a shorter time rather than using human-described features. Among all the deep learning models, U-Net architecture achieved an outstanding performance due to its capability and favourability towards biomedical image segmentation.

Thus, this project proposed a deep learning technique by using an optimised lightweight U-Net model to extract vein features from the input NIR images [51]. Handcrafted descriptor or feature extractor had no need to be defined manually because the neurons in the hidden layers of U-Net act as the feature extractor to learn vein features from the inputs.

The aim of the project was to achieve a reliable and stable performance when the model was inputted with augmented data due to small dataset size. The proposed network architecture would expand the diversity of input images using data augmentation so that the model performance will not degrade due to limited input images. A larger amount of training data would guarantee a more reliable network model along with the use of in-house collected dataset. Besides, since there were a large number of augmented images used in training phase, the original U-Net architecture was further improved into a lightweight version so that the model would not overfit the data and provide over-optimistic results.

Furthermore, multiple experiments would be carried out to test the proposed network with different parameters, including learning rates, activation functions, numbers of epochs, filter size, and number of layer blocks to find out their optimum values in order to obtain a clear predicted vein image and also to get the range of the learning rate for hyperparameter fine-tuning. In addition, unsupervised learning concept was explored to be applied in the project by comparing the chosen checkpoints between forearm images without groundtruth and the predicted outputs. This part of the whole system design has consumed the greatest amount of time and effort as it was difficult to find out a suitable method that can compare the forearm images and predicted outputs without using groundtruth. By using the selected checkpoints, true positive and true negative pixels were identified to evaluate model performance.

Chapter 3 System Model and Design

3.1 Design Specifications

3.1.1 Methodologies and General Work Procedures

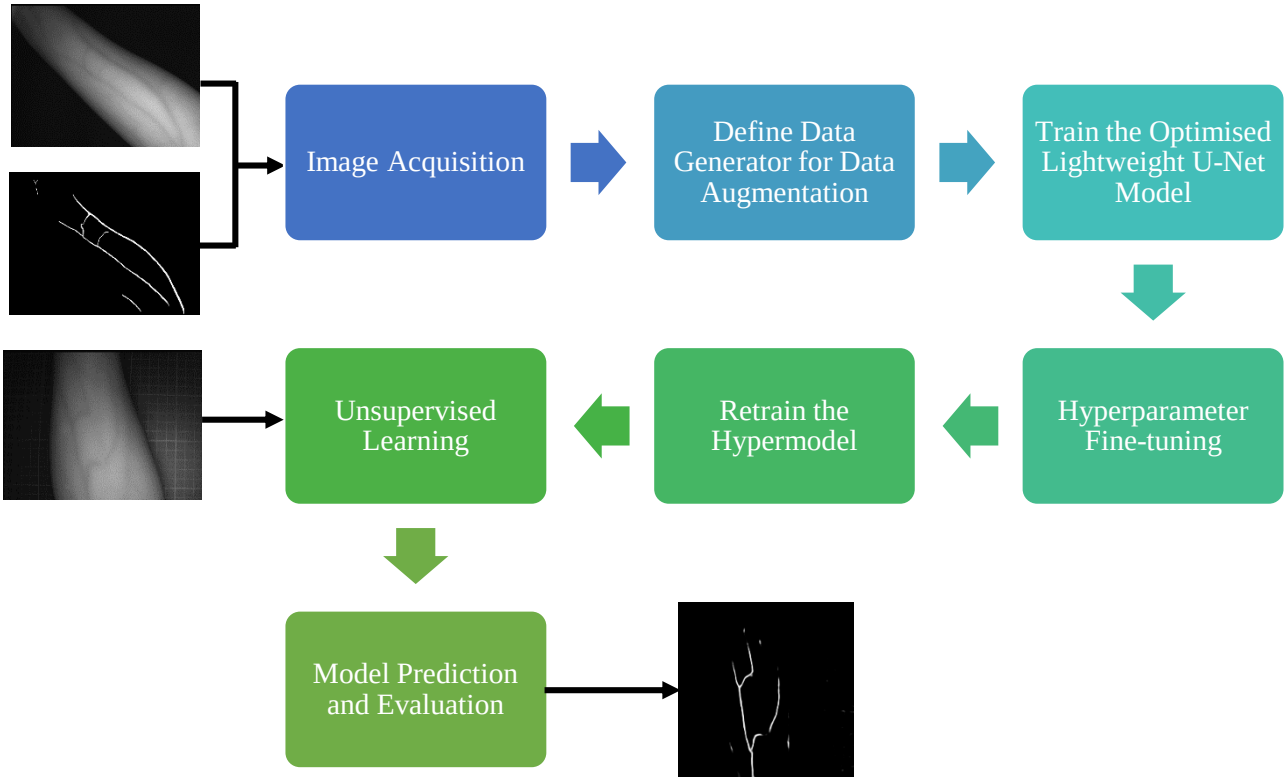


Figure 3.1 Overview flowchart of the proposed system.

An optimised lightweight U-Net model was proposed to extract subcutaneous vein features from the NIR vein images due to the robustness of the model architecture in performing biomedical image segmentation. The development of the project was categorised into different phases as illustrated in Figure 3.1, which were data acquisition, data augmentation, model construction and training, hyperparmeter fine-tuning, model retraining, unsupervised learning, and making predictions and performance evaluation using the trained hypermodel [51].

Firstly, an in-house NIR subcutaneous vein image dataset was obtained from a research group of Universiti Tunku Abdul Rahman. There was no data preprocessing being carried out since the proposed lightweight U-Net model was aimed to capture forearm spots of patients through an intelligent healthcare machine. Thus, the original images were more suitable to be used in this context. Next, a data generator dictionary

was defined to augment training images after conducting several experiments using different augmentation combinations. The data generator carried out image transformation on a real time basis using the predefined dictionary and the images were inputted into the optimised model for training. Hyperparameter fine-tuning was conducted to fine-tune learning rate so that the cost could be minimized and would reach the local minimum within a reasonable number of iterations. After retraining the hypermodel, unsupervised learning was explored by manually selecting checkpoints from random directions of forearm images from another dataset which do not have groundtruth. After making prediction on the model using the unsupervised dataset, the checkpoints selected were used to calculate performance metrics by comparing pixels of these checkpoints between the predicted output and original forearm images. The details of each phase will be discussed in upcoming sections.

3.1.2 Tools to Use

The project required both hardware devices and software to carry out and speed up the development process. The use of NVIDIA GPU has allowed parallel computation for deep learning model training as it provides support for CUDA Toolkit and cuDNN, resulting in a shorter computational time. Table 3.1 showed the hardware specification used in the project.

Table 3.1 Hardware used in this project.

Operating System	Windows 10 64-bit
Processor	AMD Ryzen 7 4800H with Radeon Graphics 2.90 GHz
RAM	16.0 GB
Graphics	NVIDIA GeForce GTX 1660 Ti 6 GB
Storage	512 GB

The software used to develop the project were listed as follows:

1. Anaconda Individual Edition

   Anaconda was used as the software to write and execute Python codes as well as to train the model. It is an open-source package and environment manager for Python and R distribution in the data science field. In the project, Python

distribution and its related packages or libraries were used. Anaconda also provides web-based interactive environments such as Jupyter Notebook and JupyterLab to ease the management of code and markdown text cells.

2. Jupyter Notebook

   Jupyter Notebook is a free web-based application that runs on a web browser to allow the writing of Python codes in notebook documents with an extension of "ipynb". It allows the project development to be carried out by writing codes and markdown texts in cells to perform data cleaning, data augmentation, ROI extraction, creation of deep learning models, result visualisation, and much more.

3. Tensorflow

   Tensorflow is an open-source machine learning framework that facilitates machine learning research and project development. It is now commonly used as one of the software libraries to build and train deep learning models using the rich collections of Python tools and libraries.

4. Keras

   Keras is an open-source high-level API to ease Python code writing. It can run as an interface on top of the Tensorflow framework. The neural network implementation of Keras includes standalone modules of network layers, activation functions, optimizers, cost functions and regularizers.

3.1.3 User Requirements

- Users shall have a basic knowledge of subcutaneous vein and its location in human skin anatomy.
- Users shall identify and verify the vein features that are successfully extracted by the proposed model.
- Users shall have a basic knowledge of IV insertion and the drawback of manual vein localization.

3.1.4 System Performance Definition

1. Accuracy

$$Accuracy = \frac{TP + TN}{TP + FP + FN + TN} \quad (1)$$

The performance of the proposed model was aimed to achieve a high accuracy in the process of vein feature extraction from NIR images, with the improvement of accuracy for at least 5% compared to any of the reviewed studies. Suitable implementations would be applied on the proposed U-Net model, such as training with different parameters, to fit the model better on the training images so that the system performance can be more accurate.

2. Confusion Matrix

		Predicted Label	
		Vein pixel not detected	**Vein pixel detected**
Real Label	**Vein pixel not present in image**	True Negative (TN)	False Positive (FP)
	Vein pixel present in image	False Negative (FN)	True Positive (TP)

Table 3.2 Confusion matrix.

A confusion matrix would be computed to show the values of four measurements as shown in the coloured table cells in Table 3.2. True positive is a measurement in which the positive vein pixels are correctly predicted as positive. False positive, also denoted as type 1 error, is the measurement in which the negative pixels are wrongly predicted as positive. False negative, also known as type 2 error, is denoted as the measurement where the positive vein pixels are wrongly predicted as negative. While true negative measures the negative vein pixels which are correctly predicted as negative. These four measurements would be used to compute another three performance scores: specificity, sensitivity, and precision.

3. Specificity

$$Specificity = \frac{TN}{TN + FP} \qquad (2)$$

Specificity is the proportion of the number of true negatives over the addition of true negatives and false positives. In other words, it denotes the probability for the model to predict a vein pixel correctly as negative given that the pixel does not contain vein features. A high specificity score means that the model is highly possible to be able to predict negative vein pixels correctly and results in a lower false positive rate.

4. Sensitivity

$$Sensitivity = \frac{TP}{TP + FN} \qquad (3)$$

Sensitivity is denoted as the proportion of true positives over the addition of true positives and false negatives. In other words, it specifies the probability for the model to predict a vein pixel correctly as positive given that the pixel contains vein features. A high sensitivity score shows that the model is highly capable to predict positive vein pixels correctly and results in a lower false negative rate.

5. Precision

$$Precision = \frac{TP}{TP + FP} \qquad (4)$$

Precision is denoted as the number of true positives over all the positives. In this context, it computes the accuracy to predict the positive vein pixels correctly. A high precision score shows that the vein pixels that are being predicted as positive actually contain vein features.

6. Binary cross entropy loss

$$BCE\ (y, \hat{y}) = -\frac{1}{m}\sum_{i=1}^{m}\left(y^i \log \hat{y}^i + \left(1 - y^i\right)\log(1 - \hat{y}^i)\right) \quad (5)$$

This loss function is most commonly used in image segmentation tasks. It carries out pixel-wise comparison between the predicted output and one-hot encoded groundtruth, and computes an average calculation over all pixels.

7. Dice coefficient

$$Dice\ Coefficient = \frac{2TP}{2TP + FP + FN} \quad (6)$$

$$Dice\ Coefficient\ Loss = 1 - \frac{2TP}{2TP + FP + FN} \quad (7)$$

Dice coefficient measures the overlap between the predicted output and the original groundtruth. A high dice coefficient score shows that there is a great similarity and overlap between the groundtruths, resulting in successful vein feature extraction. Similarly, by subtracting dice coefficient with 1, the measurement of non-overlapping pixels between the predicted output and the original groundtruth will be obtained.

3.2 System Model

3.2.1 Initial Optimized U-Net Model Architecture

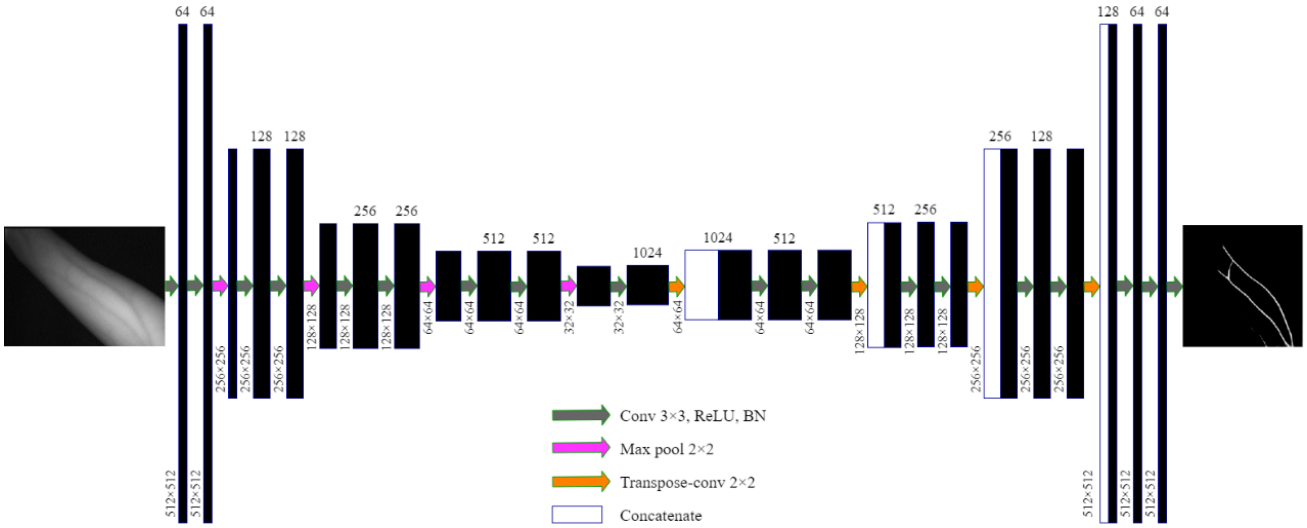


Figure 3.2 Initial optimized U-Net model architecture.

The model implemented in this project was U-Net, a fully convolutional network architecture proposed by Ronneberger et al. [52] for biomedical image segmentation. In [52], this network was implemented in ISBI cell tracking challenge in 2015 due to its robustness to segment neuronal structures from a small set of training images. Thus, the U-Net architecture was applied in this project to extract subcutaneous vein features. Initially, several optimizations were done on the existing U-Net architecture.

In the first optimization, the model consisted of four contraction blocks, the bottleneck block, and four expansion blocks. Similar to the original U-Net architecture, each contraction block contained two 3×3 convolutions followed by a 2×2 max pooling layer. In the fourth contraction block, it contained a dropout layer with a dropout ratio of 0.5. The bottleneck block contained two 3×3 convolution layers followed by a dropout layer.

For the expansion blocks, instead of using up sampling layers as in the original architecture, **3×3 transpose convolution layers (Conv2DTranspose) with stride of 2** were used to carry out convolution in an opposite direction using some parameters

rather than simply scaling up the images, as shown in Figure 3.5. The output of the transpose convolution could be computed as follows:

$$o = (i - 1) * s + k - 2p \quad (1)$$

where *o* denotes output activation size of a transpose layer, *i* denotes the input dimension, *s* denotes stride, *k* denotes filter size, and *p* denotes as padding.

In the optimized model, each expansion block contained a transpose convolution layer, a concatenate layer, a dropout layer with a dropout ratio of 0.1, and two 3×3 convolution layers. The output layer was a 1×1 convolution layer with the application of **sigmoid activation** instead of softmax activation in the original U-Net architecture. **Adam optimizer** was used instead of SGD optimizer in the original U-Net model as Adam optimizer was more robust to handle sparse gradients along with the computation of individual learning rates for different network parameters.

```
# Expansion path
dconv4 = Conv2DTranspose(512, (3, 3), strides=(2, 2), padding = 'same')(drop5)
merge_dconv4 = concatenate([drop4, dconv4], axis = 3)
drop_dconv4 = Dropout(0.1)(merge_dconv4)
dconv4 = Conv2D(512, (3, 3), activation = 'relu', padding = 'same', kernel_initializer = 'he_normal')(drop_dconv4)
dconv4 = Conv2D(512, (3, 3), activation = 'relu', padding = 'same', kernel_initializer = 'he_normal')(dconv4)
```

Figure 3.3 The use of Conv2DTranspose layer to replace UpSampling layer.

```
model.compile(optimizer = Adam(lr = 1e-4), loss = 'binary_crossentropy',
              metrics = ['accuracy', dice_coef, dice_coef_loss])
```

Figure 3.4 The use of Adam optimizer to replace SGD optimizer.

The second optimization was done on top of the first optimized model by **adding Batch Normalization layers in the contracting path**. The reason for adding Batch Normalization layers between the convolutional layers and ReLU activation layers was to standardize the output of a layer before they passed to the next block so that the covariance shift problem can be minimized and make the network training more stabilized. Through the addition of Batch Normalization layers, a more satisfied predicted output was expected to be obtained by normalizing each layer's output using the batch mean and batch variance. As a result, the current network layer would be less dependent and sensitive to the input distribution of previous layers, thus reducing the rippling effect on the signals in the consecutive layers.

```
# Contraction path
conv1 = Conv2D(64, (3, 3), activation = 'relu', padding = 'same', kernel_initializer = 'he_normal')(inputss)
conv1 = BatchNormalization(axis = 3, momentum = 0.99)(conv1)
conv1 = ReLU()(conv1)
conv1 = Conv2D(64, (3, 3), activation = 'relu', padding = 'same', kernel_initializer = 'he_normal')(conv1)
conv1 = BatchNormalization(axis = 3, momentum = 0.99)(conv1)
conv1 = ReLU()(conv1)
pool1 = MaxPooling2D(pool_size = (2, 2))(conv1)
```

Figure 3.5 The addition of Batch Normalization layers in the contraction blocks.

3.2.2 Lightweight U-Net Model Architecture

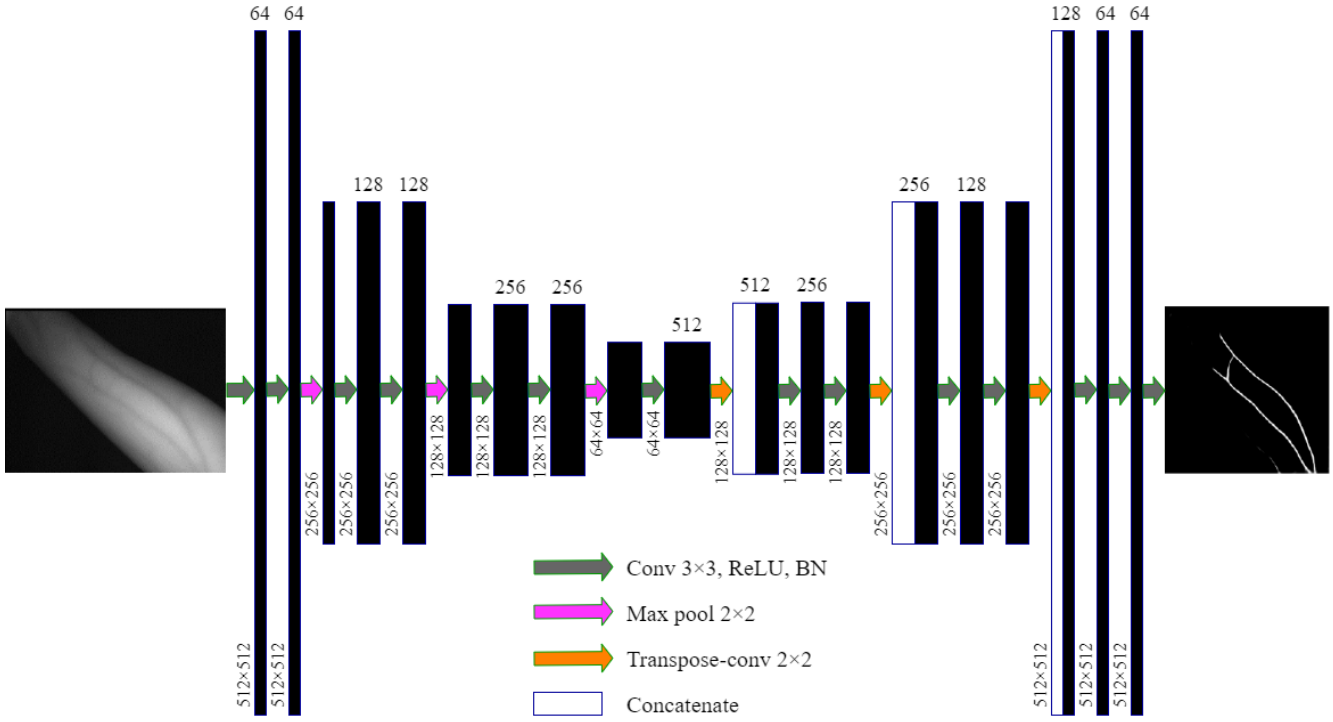


Figure 3.6 Lightweight U-Net model architecture.

After intensive model training, the optimized U-Net model was further simplified by reducing the number of network layers before hyperparameter fine-tuning as shown in Figure 3.3. The purpose of this network layer reduction was due to the observation of slight overfit on the training data. NarasingaRao et al. [53] claimed that excessive number of hidden layers might increase model architecture complexity, causing the model to "learn" too well on the training data but unable to generalize on the new unseen test data.

Thus, to lower the risk of data overfitting, the complexity of the U-Net model was reduced by having the highest filter size in the bottleneck layers as 512 instead of

1024 to trim down several hidden layers. Subsequently, the lightweight U-Net model was used for hyperparameter fine-tuning to identify the optimum learning rate and also to retrain the hypermodel.

3.3 System Design

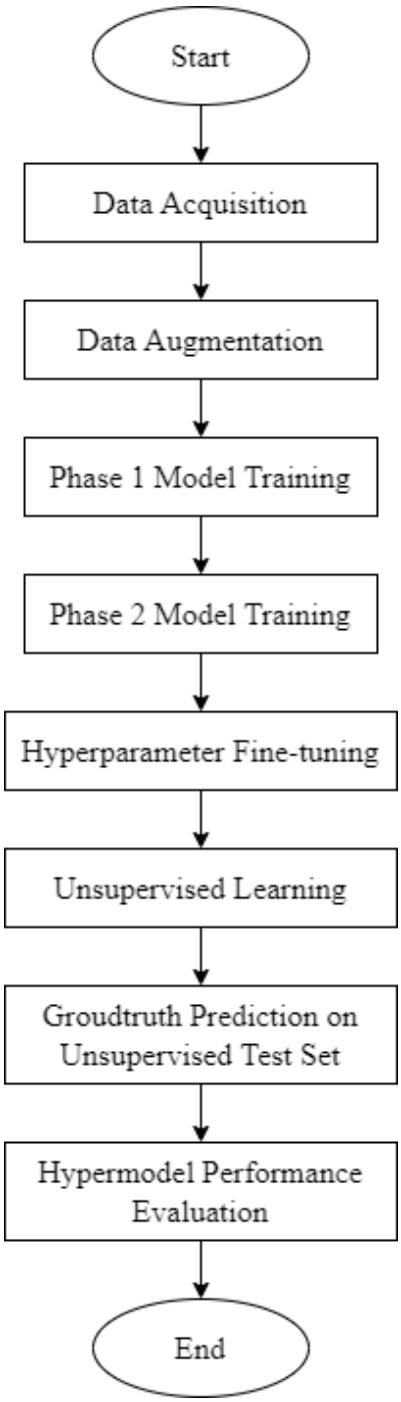


Figure 3.7 Overview flowchart of the proposed system.

3.3.1 Data Acquisition

The images obtained for this project were the forearm images and their respective groundtruth among eight subjects collected by a research group of Universiti Tunku Abdul Rahman. The in-house dataset contained eight sets of monochrome images captured using NIR camera and saved in TIF format with image size of 640 × 480 pixels. Since the objective of this project was to extract subcutaneous vein features from the input images, the images were separated into subject and mask directories. All the eight sets of subject and groundtruth images were used for training. The screenshot of the dataset images was shown in Figure 3.3 below.

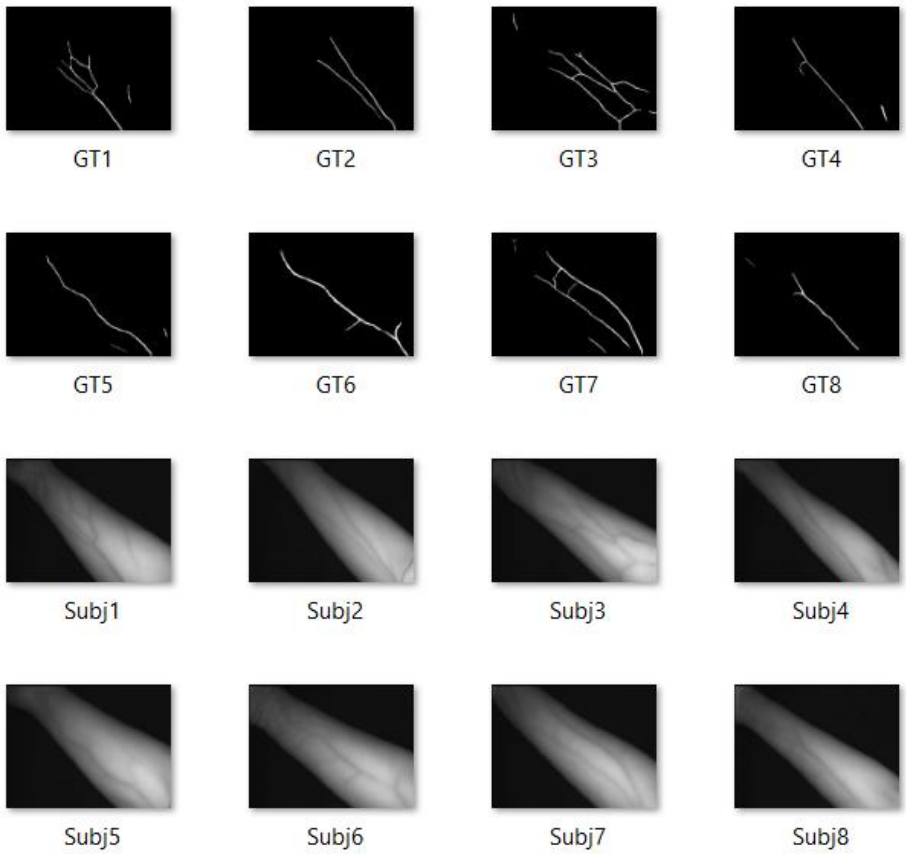


Figure 3.8 Subject forearm images and their respective groundtruth images.

3.3.2 Data Augmentation

Since the project aimed to facilitate intelligent healthcare machines to capture a patient's forearm during IV insertion, the original image was used to train the proposed model without further preprocessing or ROI extraction to better generalize to the problem statement and objectives of this project. Data augmentation was carried out to

increase the amount of training data since the dataset obtained was small as well as to avoid overfitting. Image transformation techniques in data augmentation were able to generate a larger batch of training samples from the existing images artificially and increase data diversity. The ImageDataGenerator class from Keras library was implemented to achieve data augmentation with the augmentation techniques. In this project, four different combinations of data augmentation techniques were used to conduct experiments to determine the most suitable combination of data transformation techniques. Table 3.3 showed all the tested combinations techniques and Figure 3.4 showed some of the augmented images after applying the first combination from Table 3.3. Then, the first combination of data augmentation techniques was chosen for subsequent development stages.

Table 3.3 Data augmentation techniques used.

No.	Combinations of Data Augmentation Techniques
1	rotation_range = 360, width_shift_range = 0.05, height_shift_range = 0.05, shear_range = 0.3, zoom_range = 0.3, horizontal_flip = True, vertical_flip = True, fill_mode = ‘nearest’
2	zoom_range = 0.3, shear_range = 0.3, fill_mode = ‘constant’, cval = 0., horizontal_flip = True, vertical_flip = True
3	brightness_range = [-1, 1], rotation_range = 360, zoom_range = [0.5, 1.5], horizontal_flip = True, vertical_flip = True, fill_mode = ‘constant’

No.	Combinations of Data Augmentation Techniques
4	brightness_range = [-1.5, 1.5], shear_range = 0.5, fill_mode = ‘nearest’, rotation_range = 360, width_shift_range = 0.2, height_shift_range = 0.2

Figure 3.9 Samples of augmented images after applying the first combination of data augmentation techniques.

3.3.3 Phase 1 Model Training

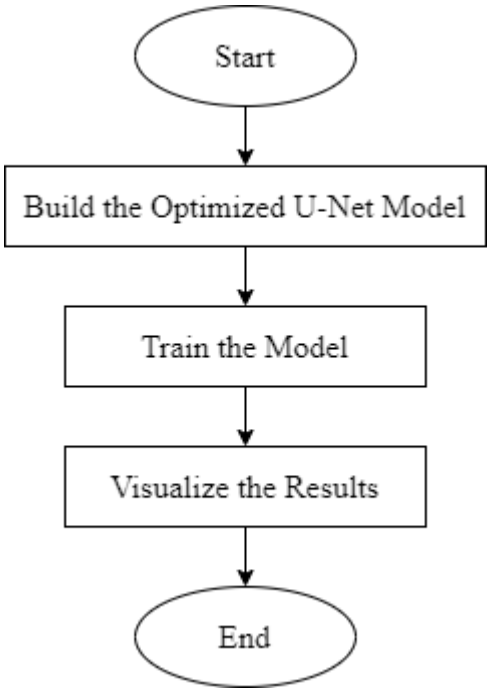


Figure 3.10 Flowchart of phase 1 model training.

During the first phase of model training, U-Net model, a fully convolutional network architecture proposed in [52] was used as the baseline model for further optimization using Python programming language, Keras library, and Tensorflow library. The up-sampling layers in the expansion blocks were replaced with **3 × 3 transpose convolution layers (Conv2DTranspose) with stride of 2** with each expansion block contained a transpose convolution layer, a concatenate layer, a dropout layer with a dropout ratio of 0.1, and two 3 × 3 convolution layers. In the output layer, which is a 1 × 1 convolution layer, softmax activation in the original U-Net architecture was replaced with **sigmoid activation**. In terms of optimizer, SGD optimizer was replaced with **Adam optimizer** as it was more robust in handling sparse gradients besides having a faster convergence. While in the contracting path, **Batch Normalization layers** were added to minimize the effect of covariance shift problem between the hidden layers. The pseudocode of the first optimized model architecture was shown in Table 3.4.

Table 3.4 Pseudocode of constructing the first optimized model architecture.

```
Pseudocode: Optimized U-Net architecture
    Objectives:
    1   improve the original U-Net model to suit better on vein segmentation task.
    Inputs:
    1   original U-Net architecture.
    2   original hyperparameter configurations.
    Outputs:
    1   optimized U-Net architecture.
    Coding:
        Build the original U-Net architecture with original hyperparameter settings.
        Change UpSampling layers to Conv2DTranspose layers in the expansion path.
        Replace softmax activation of the output layer with sigmoid activation.
        Replace SGD optimizer with Adam optimizer.
        Add another two metrics (dice coefficient and dice coefficient loss).
        Remove ReLU activation in the convolution layers.
        Add BatchNormalization layers after every convolution layer.
        Add ReLU layers after the BatchNormalization layers.
```

After constructing the model architecture, several training experiments were conducted to determine the most suitable data augmentation combination as well as a suitable set of hyperparameters including learning rate, activation function, and number of epochs. The training process was divided into three parts to compare between optimized model architecture, data augmentation techniques, and hyperparameters.

Firstly, four combinations of data augmentation techniques were declared using the training generator class defined by ImageDataGenerator. All the training generators contained training and validation subsets with a validation split of 0.3. Each data generator was being inputted to the training process to undergo real time augmentation during model training so that the model can train on different batches of augmented images in each epoch.

During the real time data augmentation, the images had undergone three preprocessing steps in the background after the augmentation techniques were applied to them. Firstly, each augmented image, including the forearm images and groundtruth

images, were resized into 256×256 pixels. Secondly, the images were normalized by dividing the pixels with 255 so that the pixels ranged between 0 and 1. Thirdly, groundtruth images were one-hot encoded using a predefined threshold to assign vein pixels with "1" and non-vein pixels with "0". All these steps were carried out prior to training the images on the model. After completing a training process, the trained models were saved into model checkpoints. The training pipeline was carried out in a sequence as shown in Table 3.5 below.

Table 3.5 Phase 1 training process.

Model Version	**Comparison Aspects**	**Epochs**	**Steps per Epoch**	**Validation Steps**
A. Comparison Between Optimized Network Architecture				
1	Replaced UpSampling layers with **Conv2DTranspose layers**.	5	1000	
2	Added **Conv2DTranspose layers** and **BatchNormalization layers**.	5		
B. Comparison Between Data Augmentation Techniques				
1	Combination 1	5	1000	
2	Combination 2	5		
3	Combination 3	5		
4	Combination 4	5		
C. Comparison Between Hyperparameters				
1	learning rate = 1e-3	5	1000	
2	learning rate = 1e-4	5		
3	learning rate = 1e-5	5		
4	activation = 'relu'	5	1000	
5	activation = 'elu'	5		
6	epochs = 5	5	1000	
7	epochs = 10	10		
8	epochs = 15	15		
9	epochs = 25	25		

During the training process, there were four performance metrics being monitored, which were binary cross entropy loss, accuracy, dice coefficient, and dice coefficient loss. The performance metrics for each epoch were saved in the history callback dictionary. Scores including accuracy, loss, and dice coefficient were used for plot visualization to observe some insights about the training process and will be discussed in the following chapter.

```
print(training_ver1.history.keys())
```

```
dict_keys(['loss', 'accuracy', 'dice_coef', 'dice_coef_loss', 'val_loss', 'val_accuracy', 'val_dice_coef', 'val_dice_coef_los
s'])
```

Figure 3.11 The performance metrics saved in the history callback.

3.3.4 Phase 2 Model Training

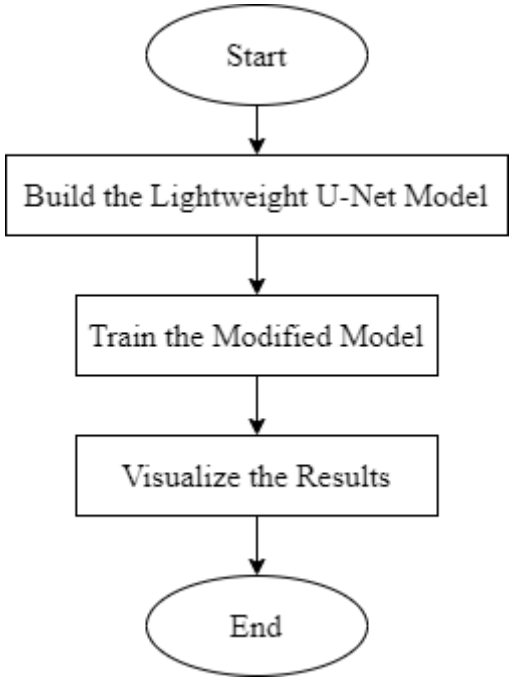


Figure 3.12 Flowchart of phase 2 model training.

The second phase of model training focused on building a more improved U-Net model architecture that could give a lower data overfitting risk. The U-Net model in the first phase was further optimized by reducing the number of hidden layers. In order to determine the most suitable number of layer blocks and filter sizes used in the blocks, several experiments had been conducted to train different versions of lightweight model as shown in Table 3.6 along with the specifications given.

Table 3.6 Phase 2 training process.

Lightweight U-Net Version	Number of Layer Blocks	Filter Size	
		First Convolution Block	Bottleneck Block
1	8	64	512
2	6	64	256
3	10	16	256
4	8	16	128

During the model training, all the hyperparameters used in the .fit command, which were epochs, steps per epoch, validation steps, and also the data augmentation generator, were similar as determined from phase 1 model training. From the training process, the first lightweight U-Net version obtained the highest dice coefficient besides successfully reducing the network complexity. The risk of data overfitting was observed by visualizing the three performance metrics, which were accuracy, binary cross entropy loss, and dice coefficient. The first lightweight U-Net version was used for the subsequent steps.

Table 3.7 Pseudocode of constructing the optimized lightweight model architecture.

Pseudocode: Optimized lightweight U-Net architecture

Objectives:

1 improve the overfitting risk of the optimized U-Net model in Phase 1 model training.

Inputs:

1 optimized U-Net architecture.

2 the chosen data augmentation combination and hyperparameter values.

Outputs:

1 optimized lightweight U-Net architecture.

Coding:

Build the optimized U-Net architecture with chosen data augmentation combination and hyperparameter configurations.

Reduce the convolutional layer blocks.

Define the initial filter size and subsequent filter sizes according to the number of layer blocks.

Repeat the steps above to define another three lightweight models to determine the most suitable number of layer blocks and filter size for the layers.

3.3.5 Hyperparameter Fine-Tuning

```
tuner = kt.Hyperband(hyper_unet,
                     objective = kt.Objective("dice_coef", direction="max"),
                     max_epochs = 30,
                     project_name = 'unet_finetune')
```

Figure 3.13 Definition of hyperparameter tuner.

In hyperparameter fine-tuning stage, Keras Tuner library was used to determine the optimum hyperparameter defined in the model. Prior conducting the hypertuning, a Hyperband tuner was selected among other tuners and was instantiated. The reason of choosing Hyperband tuner was that it allowed the use of early stopping when searching for optimum hyperparameter values to speed up the convergence on the model and stop the training before the model overfits the data.

```
kt_learning_rate = kt.Float('steps',
                            min_value = 0.0001,
                            max_value = 0.001,
                            step = 0.00005)

model.compile(optimizer = Adam(lr = kt_learning_rate),
              loss = 'binary_crossentropy',
              metrics = ['accuracy', dice_coef, dice_coef_loss])
```

Figure 3.14 Definition of hyperparameter search space for learning rate.

The hypermodel with the name “hyper_unet” was defined to create a hyperparameter search space that allows the tuner to search for an optimum value within the defined space. Figure 3.14 above showed the definition of search space for learning rate, in which the learning rate would be tuned between 0.0001 and 0.001 with the increment or decrement of 0.00005 during each hyperparameter search.

The objective of the tuner was dice coefficient, which meant that the tuner would optimize dice coefficient towards its maximum, or the highest value possible. The maximum epoch to train a model was set as 30, and the files generated during the hyperparameter fine-tuning trials would be saved in the directory named as “unet_finetune”.

```
stop_early = tf.keras.callbacks.EarlyStopping(monitor = 'dice_coef_loss',
                                              patience=5)
```

Figure 3.15 Definition of early stopping.

Figure 3.15 showed the definition of an early stopping callback. The defined callback would monitor dice coefficient loss. If the dice coefficient loss has no improvement for 5 epochs, the training would be stopped. The reason of defining patience value was to add a delay for the end of training. This was because the first indication of no improvement might give another meaning that the model might coast into a plateau. There was a possibility where the performance would hike up again or the performance might drop before getting further improvements.

```
tuner.search(myGene_1,
             steps_per_epoch = 1000,
             epochs = 25,
             validation_data = myGene_1,
             validation_steps = 1000,
             callbacks=[stop_early])

# Get the optimal hyperparameters
best_hps=tuner.get_best_hyperparameters(num_trials=1)[0]
```

Figure 3.16 Codes to run hyperparameter search.

After instantiating the tuner and early stopping callback, the hyperparameter search was ready to be started by executing tuner.search command. The arguments in this command were the same as the arguments used in .fit command during model training, but the original callback was replaced with the early stopping callback.

The hyperparameter search would run for a number of trials. After the searching was complete, the best set of hyperparameters would be obtained. Then, the optimum hyperparameter, which was the learning rate would be used to reinstantiate and retrain the hypermodel.

3.3.6 Unsupervised Learning

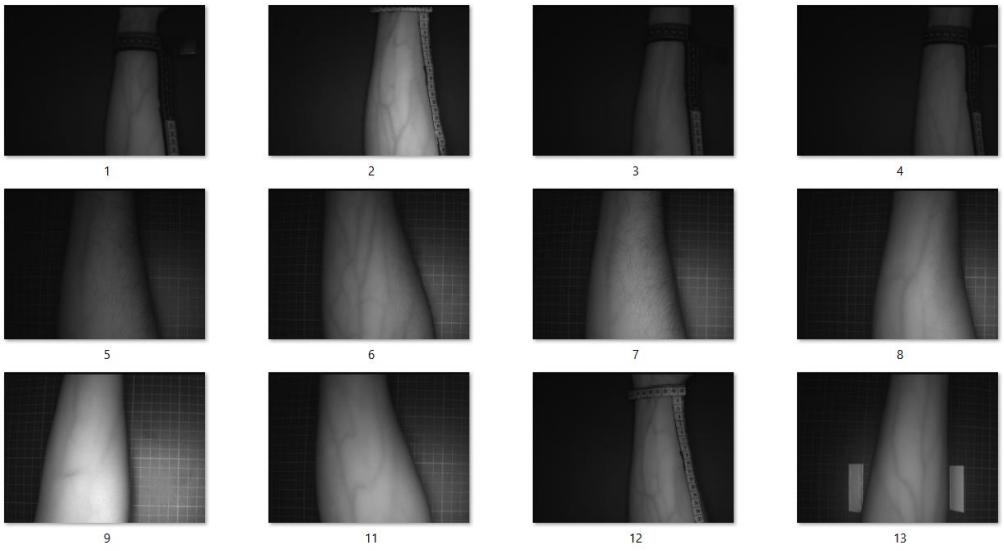


Figure 3.17 Sample images in the unsupervised dataset.

In model prediction phase, an unsupervised dataset without groundtruth collected by a research group of Universiti Tunku Abdul Rahman, with some sample images shown in Figure 3.17, was used to explore unsupervised subcutaneous vein segmentation using the proposed lightweight U-Net model. The in-house unsupervised dataset contained 31 forearm monochrome images saved with TIF format with the image dimension of 640 × 480 pixels.

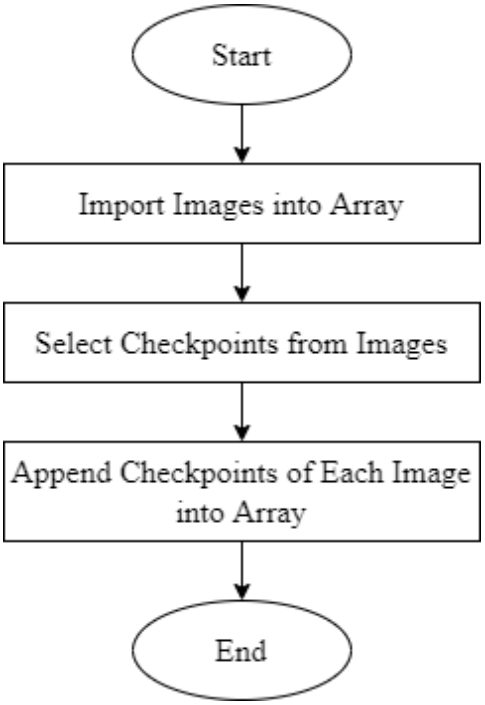


Figure 3.18 Flowchart of unsupervised subcutaneous vein segmentation.

Figure 3.18 above showed the steps carried out for unsupervised subcutaneous vein segmentation. Firstly, the unlabelled images were imported into an array. Each forearm image was then shown in a window displayed with OpenCV library that could enable the selection of checkpoints through mouse click via a self-defined function, click_event as shown in Figure 3.19. Using this function, 10 true vein checkpoints, which were the coordinates where the pixel contained vein feature, were selected for 1 forearm image from random directions. When the mouse was clicked on a particular point, a coordinate would be displayed on the image as shown in Figure 3.20, and appended into an array for later use. After selecting checkpoints for all the 31 images, the array contained 31 arrays with each arrays having 10 coordinates. The steps were repeated for negative pixels, in which the pixels contained no vein features but miscellaneous features such as the background, forearm fur, wrist tie, tourniquet, or ruler. Since checkpoint selection was a dominant step for comparison of true and false vein pixels between forearm images and predicted output, it had consumed the largest amount of time and effort as well.

```python
# function to display the coordinates of the points clicked on the image
def click_event(event, x, y, flags, params):

    fontScale = 0.5

    # checking for left mouse clicks
    if event == cv2.EVENT_LBUTTONDOWN:

        # displaying the coordinates on the notebook
        # print(x, ' ', y)

        # displaying the coordinates on the image window
        font = cv2.FONT_HERSHEY_SIMPLEX
        cv2.putText(img, str(x) + ',' + str(y), (x,y), font, fontScale, (255, 0, 0), 2)
        cv2.imshow('image', img)

        coor_per_img.append((x, y))
```

Figure 3.19 Screenshot of click_event function.

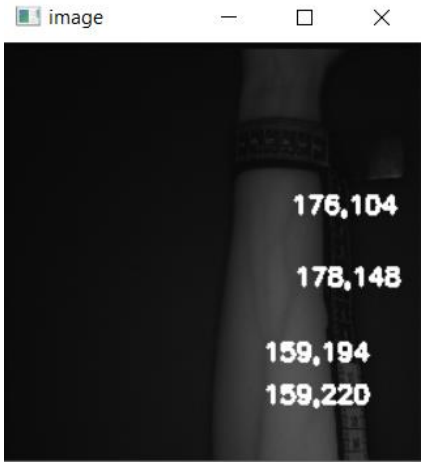


Figure 3.20 An example of selecting checkpoints from a forearm image.

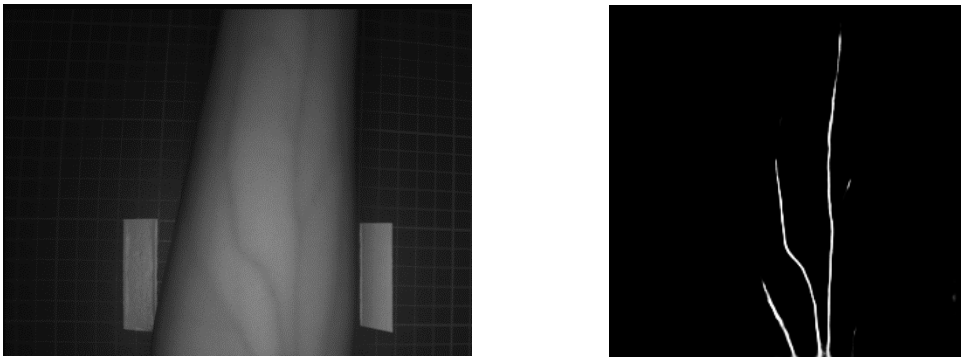
```python
# positive pixels

coor_P_file = "coor_P.pkl"

open_coor_P_file = open(coor_P_file, "wb")
pickle.dump(coor_total_P, open_coor_P_file)
open_coor_P_file.close()

read_coor_P_file = open(coor_P_file, "rb")
loaded_coor_P = pickle.load(read_coor_P_file)
read_coor_P_file.close()

print(loaded_coor_P)
```

Figure 3.21 Code to save the true vein checkpoints using pickle library.

After completing checkpoint selection for both true vein pixels and negative pixels, the arrays that contained the checkpoints were saved in .pkl files using pickle library. The code to save true vein checkpoints was shown in Figure 3.21 above. The purpose of saving the coordinates in local directory was to prevent the laborious work needed to be done to choose the checkpoints every time the checkpoints were needed for model prediction and performance evaluation. By saving the checkpoints in local directory, it could simply be read and loaded anytime when any operation that would require the use of checkpoints was to be conducted.

3.3.7 Model Prediction on Unsupervised Test Set

The unsupervised dataset was used to generate predicted outputs using the hypermodel. After resizing the images into a dimension of 256 × 256, the presaved hypermodel weight checkpoint was loaded and used to predict on the forearm images using .predict command. The predicted outputs were saved into local directory for performance metrics calculation. Figure 3.22 below showed one of the forearm images with its relative predicted output.



Unsupervised Forearm Image

Predicted Output

Figure 3.22 An example of forearm image and its predicted output.

3.3.8 Hypermodel Performance Evaluation

Performance evaluation was carried out by comparing the saved checkpoints between the forearm images and the predicted output. Firstly, the predicted outputs

were loaded from local directory into an array. Next, one-hot encoding was performed on the predicted outputs by setting the pixel that was more than 0.5 as 1, indicating a positive vein pixel, and the pixel that was equal or less than 0.5 as 0, indicating a negative pixel. The four measures (TN, TP, FP, and FN) were determined using the positive and negative checkpoint arrays in the following conditions:

- If the checkpoint value in the positive vein checkpoint array was greater than or equal to 0.5, then it was counted as TP. If the value was less than 0.5, it was counted as FN.
- If the checkpoint value in the negative vein checkpoint array was greater than or equal to 0.5, then it was counted as FP. If the value was less than 0.5, it was counted as TN.

After completing the count accumulation of TP, FN, FP, and TN, the values were used to calculate performance metrics including dice coefficient, global accuracy, sensitivity, specificity, and precision for performance discussion and analysis.

3.4 Implementation Issues and Challenges

There were several challenges encountered during the development of this project.

1. **Hardware resource limitation**

   Since this project required training on numerous epochs and real time augmentation, GPU memory was essential to speed up the training time. However, GPU computing resources were not provided by my own laptop. Training on CPU memory had prolonged the training time and slowed down the development progress. Google Colab also had limited RAM and GPU memory which caused restrictions in training large batches of augmented data. Although Google Colab offered resource upgrades, the subscription fee was expensive in the long term. Thus, this project was developed using a borrowed high-end gaming laptop that contained a decent GPU to speed up the training progress.

2. **Insufficient training dataset**

   There were many vein datasets available from different sources. However, most of them were finger vein datasets collected for biometric verification purpose. Including biomedical image collection and cleaning into the planning of project development would consume a massive amount of time and financial budget. Thus, a small in-house dataset was obtained from the university for model training purpose and another small unsupervised dataset for model evaluation purpose so that more time and effort would be focused on the project development. Although the in-house datasets used in this project had better quality compared to open source datasets, the dataset sizes were small and inadequate to build a prediction model that can be deployed in real world medical application. Hence, this project had tried to overcome this limitation by implementing data augmentation in the training images to generate large batches of images from the existing images. While for the unsupervised test images, no augmentation techniques were applied and the dataset size was remained.

3.5 Timeline

Figure 3.23 and Figure 3.24 showed the Gantt charts for the whole project development timeline and milestones. The duration of the whole project was 2 trimesters, which were approximately 26 weeks.

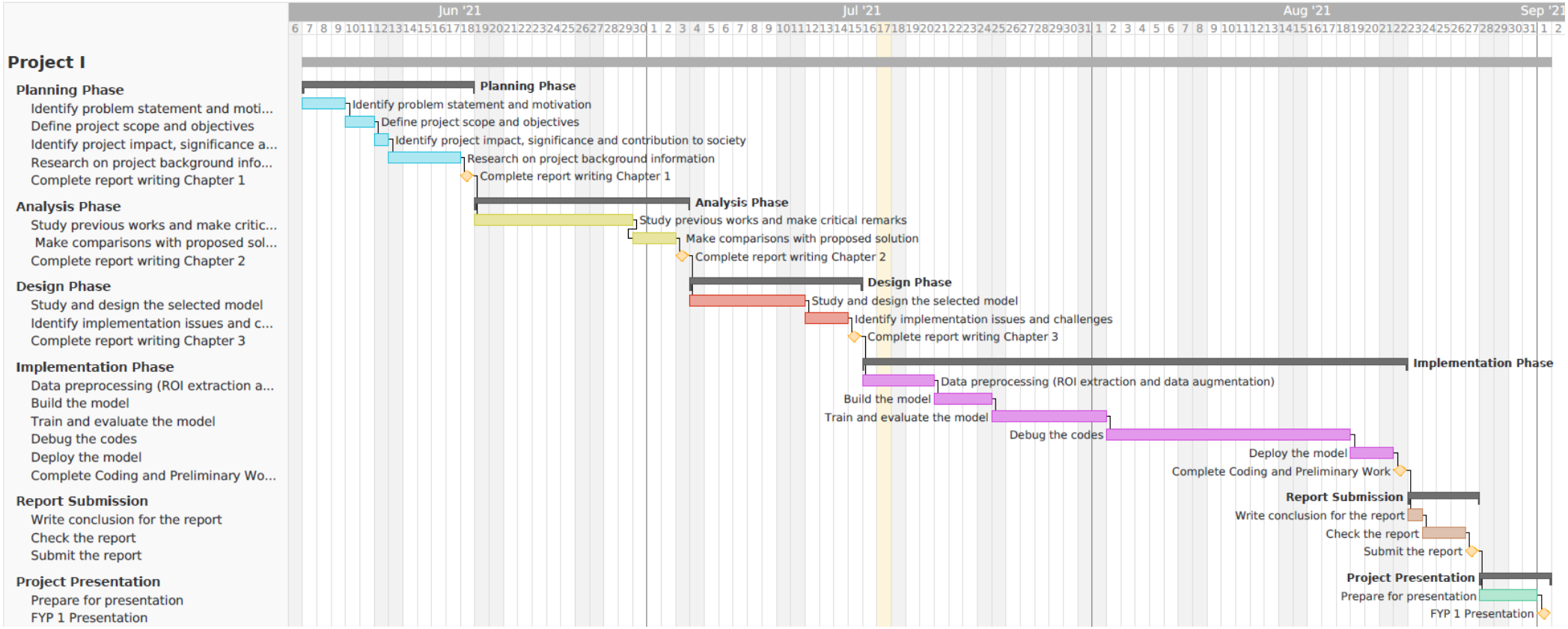


Figure 3.23 Gantt chart for the project development timeline (June to September 2021 trimester).

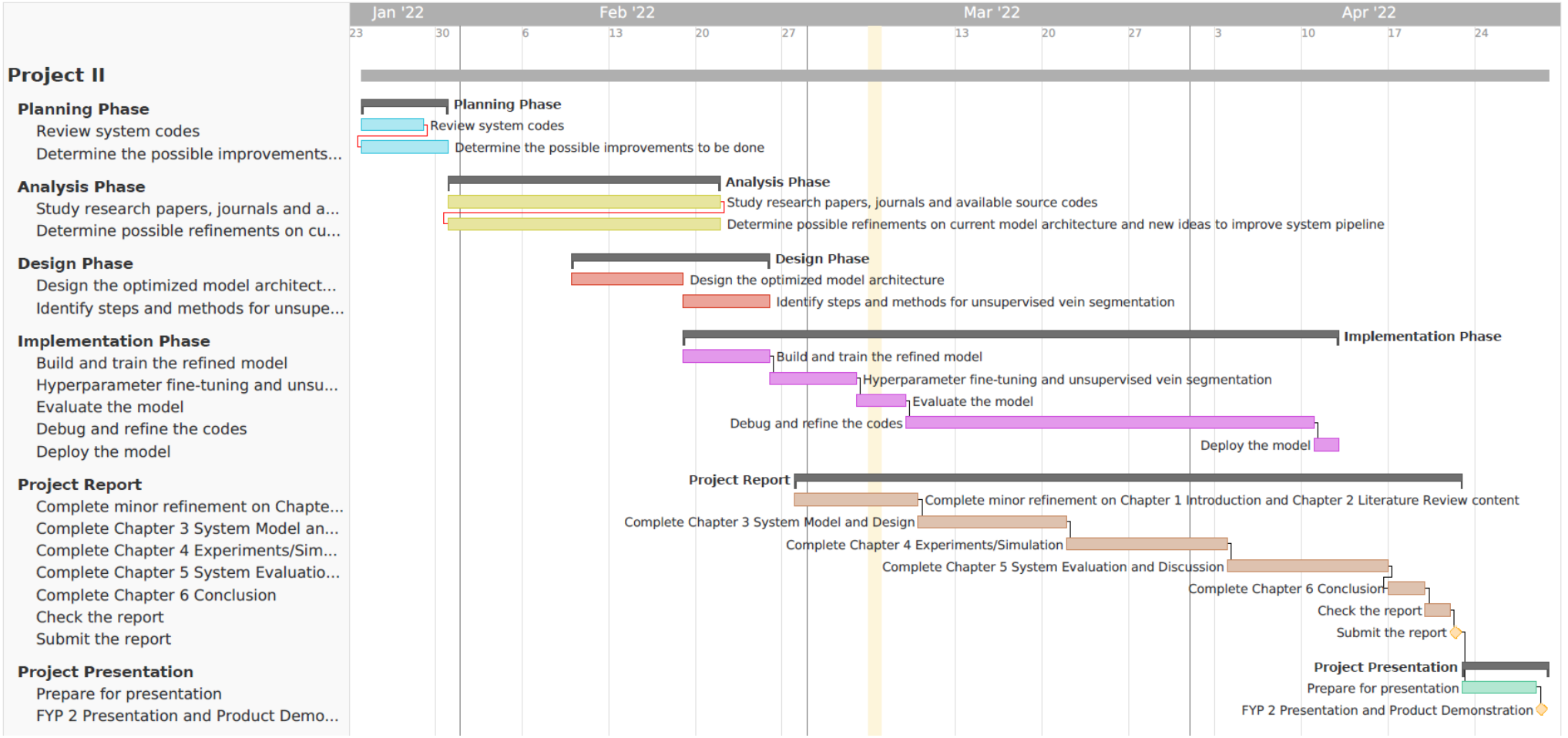


Figure 3.24 Gantt chart for the project development timeline (January to April 2022 trimester).

Table 3.8: Milestones of the project report writing in current semester and its key deliverables.

No.	Milestone	Key Deliverable(s)	Date to be Completed
1	Week 7		
	Complete the report writing for Chapter 1 and Chapter 2.	1) Modified project introduction, project scope, objectives, and literature review.	9 March 2022 (Wed)
2	Week 9		
	Complete the report writing for Chapter 3.	1) Refined system model and design.	21 March 2022 (Mon)
3	Week 11		
	Complete the report writing for Chapter 4.	1) Detailed explanation of experiments conducted.	3 April 2022 (Sun)
4	Week 12		
	Complete the model construction and coding implementation of the system. Complete the report writing for Chapter 5.	1) Full Python codes of the proposed model and the whole system. 2) Detailed explanation and analysis of the results as well as reasonable assumptions.	16 April 2022 (Sat)
5	Week 13		
	Report submission.	1) Project report.	22 April 2022 (Fri)
6	Week 14		
	Complete the project presentation and product demonstration.	1) Presentation slide and video. 2) Demonstration of proposed system.	29 April 2022 (Fri)

Table 3.9: Planning for the previous semester and current semester.

No.	Milestone	Week to be Completed
Previous semester		
1	Identify project objective	1
2	Study previous works	4
3	Determine and study a suitable model	5
4	Customize the selected model	6
5	Train and evaluate the model	7
6	Debug the codes	11
7	Deploy the model	11
8	Report submission	12
9	Presentation preparation	13
Current semester		
1	Further study the possible improvements	1
2	Refine the design of the current model architecture	4
3	Design the pipeline for unsupervised subcutaneous vein segmentation	5
4	Build and train the model	5
5	Hyperparameter fine-tuning and subcutaneous unsupervised vein segmentation	6
6	Evaluate the model	7
7	Debug and refine the model	11
8	Test and deploy the model	12
9	Report submission	13
10	Presentation preparation	14

Chapter 4 Experiments and System Evaluation

4.1 Phase 1 Model Training

The phase 1 model training of the project focused primarily on building and optimizing the existing U-Net model for subcutaneous vein feature extraction and segmentation. After finalizing the optimized network architecture, the model was trained on different sets of data augmentation techniques to find out the most suitable combination as well as some hyperparameters to determine their suitable values. Figure 4.1 showed the information generated during model training.

```
Epoch 1/5
1000/1000 [==============================] - 366s 339ms/step - loss: 0.0373 - accuracy: 0.9901 - dice_coef: 0.1979 - dice_coef_
loss: 0.8021 - val_loss: 0.0180 - val_accuracy: 0.9923 - val_dice_coef: 0.4151 - val_dice_coef_loss: 0.5849

Epoch 00001: loss improved from inf to 0.03733, saving model to unet_A1.hdf5
Epoch 2/5
1000/1000 [==============================] - 281s 281ms/step - loss: 0.0159 - accuracy: 0.9931 - dice_coef: 0.4648 - dice_coef_
loss: 0.5352 - val_loss: 0.0151 - val_accuracy: 0.9933 - val_dice_coef: 0.4897 - val_dice_coef_loss: 0.5103

Epoch 00002: loss improved from 0.03733 to 0.01590, saving model to unet_A1.hdf5
Epoch 3/5
1000/1000 [==============================] - 279s 279ms/step - loss: 0.0143 - accuracy: 0.9937 - dice_coef: 0.5171 - dice_coef_
loss: 0.4829 - val_loss: 0.0135 - val_accuracy: 0.9940 - val_dice_coef: 0.5248 - val_dice_coef_loss: 0.4752

Epoch 00003: loss improved from 0.01590 to 0.01426, saving model to unet_A1.hdf5
Epoch 4/5
1000/1000 [==============================] - 282s 282ms/step - loss: 0.0133 - accuracy: 0.9941 - dice_coef: 0.5507 - dice_coef_
loss: 0.4493 - val_loss: 0.0126 - val_accuracy: 0.9944 - val_dice_coef: 0.5759 - val_dice_coef_loss: 0.4241

Epoch 00004: loss improved from 0.01426 to 0.01328, saving model to unet_A1.hdf5
Epoch 5/5
1000/1000 [==============================] - 281s 281ms/step - loss: 0.0121 - accuracy: 0.9946 - dice_coef: 0.5893 - dice_coef_
loss: 0.4107 - val_loss: 0.0111 - val_accuracy: 0.9951 - val_dice_coef: 0.6239 - val_dice_coef_loss: 0.3761

Epoch 00005: loss improved from 0.01328 to 0.01213, saving model to unet_A1.hdf5
```

Figure 4.1 The information generated during the training process.

```
image_datagen = ImageDataGenerator(**aug_dict, validation_split = 0.3)
mask_datagen = ImageDataGenerator(**aug_dict, validation_split = 0.3)
```

Figure 4.2 Training and validation subset splitting using a ratio of 7:3.

From Figure 4.1, the green box showed the number of steps in an epoch, which was 1000. This indicated that the augmented images generated by the ImageDataGenerator were divided into 1000 batches during the training process with each batch being trained on different images augmented on the fly. By defining training and validation subsets for ImageDataGenerator in 7:3 ratios as shown in Figure 4.2, the scores for losses and accuracies were generated for the two subsets. The accuracy score referred to the model accuracy when it was trained on the training subset, while

validation accuracy denoted the accuracy to predict on validation subset during model validation. Similarly, loss indicated the error incurred by the model during training, while validation loss indicated the error incurred during model validation. By observing these metrics, overfitting can be detected.

However, the model had a high accuracy and low loss from the beginning of model training, which might mislead the real interpretation of model performance. The reason for these extremely optimistic metric scores was that almost 99% of all the pixels were zero since vein pixels only held a small proportion of pixels in an image. Getting a high accuracy and low loss from the start was trivially easy as the model was able to classify most of the zero value pixels. Thus, these metrics might not be meaningful to determine the real performance of the model and were used mainly to observe overfitting in term of graph visualization.

Dice coefficient and dice coefficient loss, as circled in red boxes, were more suitable to be used as the indication of model performance in this project. This was because the dice coefficient measured the number of positive pixels that can be determined by the model to detect similarity between predicted outputs and the gorundtruths. A high dice coefficient indicated that the model was able to predict most of the positive pixels correctly and had a great similarity with the groundtruths, while a low dice coefficient meant that the model was not able to identify positive pixels from the forearm images.

4.2 Analysis of Model Training on Optimized Network Architecture

To obtain the best network architecture, the data generator defined earlier was first trained on the optimized U-Net that used Conv2DTranspose layers instead of original UpSampling layers in the expansion path. The dice coefficient scores and the predicted output were shown in Table 4.1 and Figure 4.3 respectively.

Table 4.1 Dice coefficient scores for the first optimized network architecture.

Epoch	Training Dice coefficient	Validation Dice Coefficient
1	0.1979	0.4151
2	0.4648	0.4897
3	0.5171	0.5248
4	0.5507	0.5759
5	0.5893	0.6239

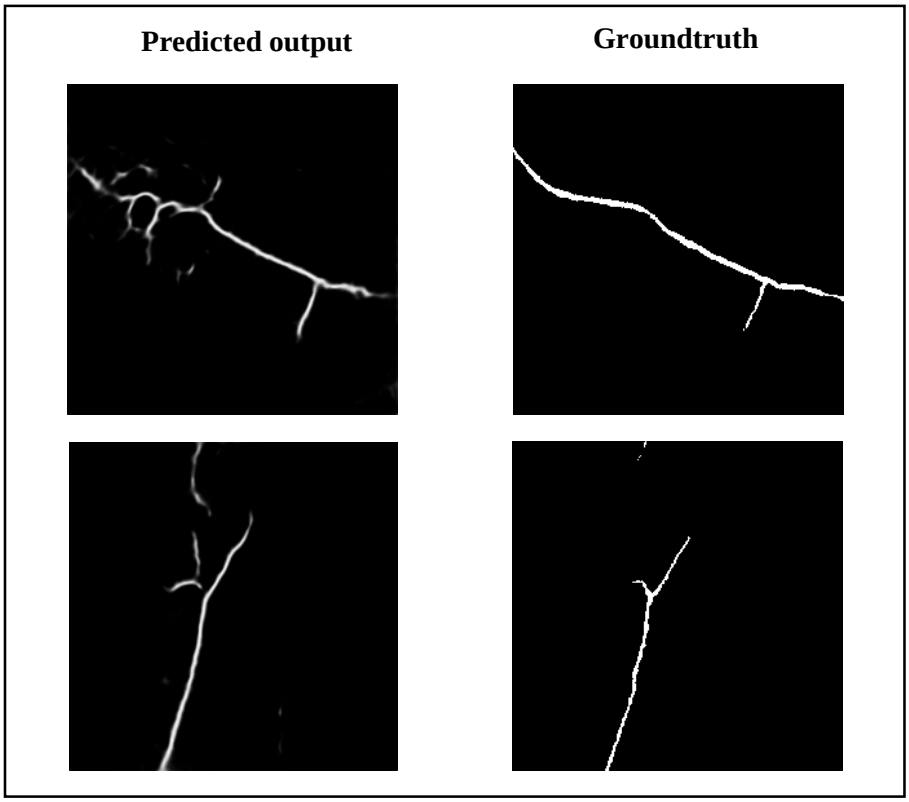


Figure 4.3 Example of predicted outputs for the first optimized network architecture.

As observed in Figure 4.3, the predicted output of the first optimized U-Net architecture contained noise in which the forearm textures were being wrongly extracted as vein features. The low dice coefficient score in Table 4.1 indicated that this optimized model was less powerful to identify positive pixels correctly, resulting in a smaller overlap of vein pixels between predicted output and groundtruth.

The model was further optimized by adding BatchNormalization layers between convolution layers and activation layers to standardize the input before entering subsequent convolutions. Batch normalization can maintain the input with variance of

1 and mean of 0, resulting in less covariance shift issue and reduced the sensitivity of the current layer towards the variations in input distribution. Table 4.2 and Figure 4.4 showed the dice coefficient scores generated and the predicted outputs respectively.

Table 4.2 Dice coefficient scores for the second optimized network architecture.

Epoch	Training Dice coefficient	Validation Dice Coefficient
1	0.3799	0.4747
2	0.5477	0.5667
3	0.6049	0.5558
4	0.6436	0.6583
5	0.6764	0.7032

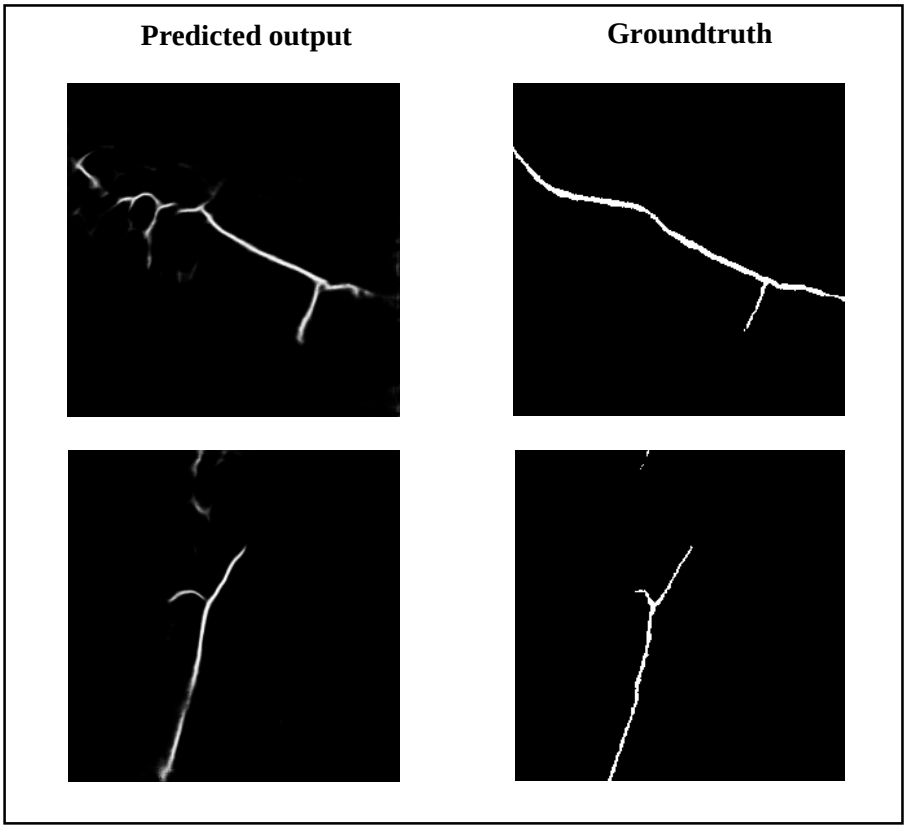


Figure 4.4 Examples of predicted outputs for the second optimized network architecture.

The dice coefficient score obtained was higher than the previous optimized network architecture, which meant that the second optimized model was more robust towards input variation with the implementation of batch normalization. From Figure 4.4, it was observed that less noise was present in the predicted output, especially for

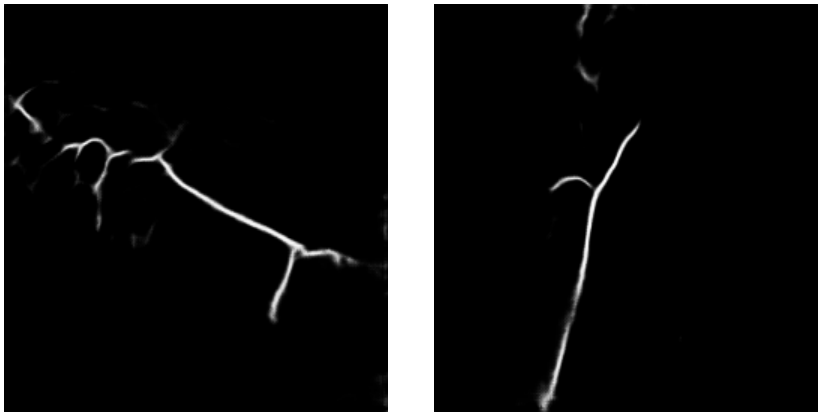
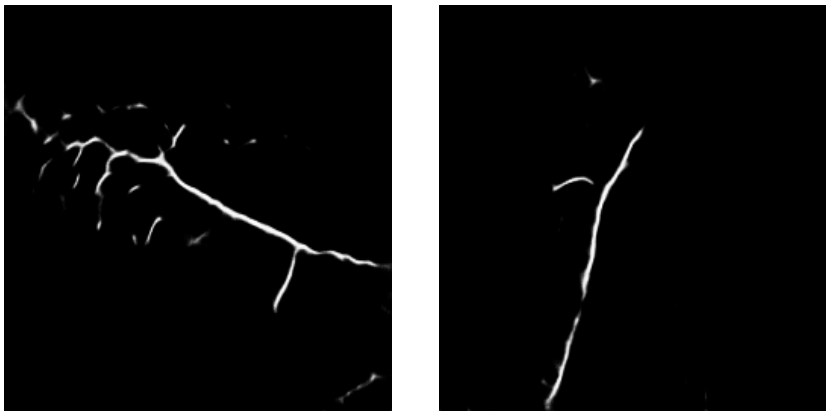
the second example. This has indicated that the vein pixels were able to be predicted more precisely compared to the previous optimized version.

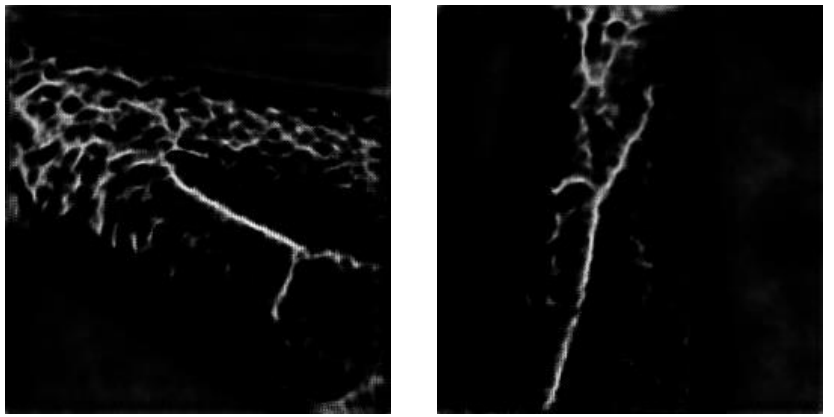
4.3 Analysis of Model Training on Data Augmentation

After training the optimized model architecture, the predefined data augmentation combinations were applied to transform the training set of both vein and groundtruth images to investigate the dice coefficient scores generated as well as their predicted outputs. The purpose of investigating different augmentation techniques was to obtain the best data augmentation combination that can extract vein features with the least noise while at the same time, retain most of the useful vein features.

Since the optimized network architecture training in the previous section utilized the first combination during its training, another three training processes were conducted for combination 2, 3, and 4. Similarly, these augmentation dictionaries were used in ImageDataGenerator to perform real time augmentation during model training. Table 4.3 listed the dice coefficient scores achieved by each data augmentation combination along with the predicted output.

Table 4.3 Dice coefficient scores and predicted outputs for each data augmentation combination.

No.	Combinations of Data Augmentation Techniques	Dice Coefficient Scores Obtained		Examples of Predicted Outputs
		Training	Validation	
1	rotation_range = 360, width_shift_range = 0.05, height_shift_range = 0.05, shear_range = 0.3, zoom_range = 0.3, horizontal_flip = True, vertical_flip = True, fill_mode = 'nearest'	0.6764	0.7032	
2	zoom_range = 0.3, shear_range = 0.3, fill_mode = 'constant', cval = 0., horizontal_flip = True, vertical_flip = True	0.8100	0.7895	

No.	Combinations of Data Augmentation Techniques	Dice Coefficient Scores Obtained		Example of Predicted Outputs
		Training	Validation	
3	brightness_range = [-1, 1], rotation_range = 360, zoom_range = [0.5, 1.5], horizontal_flip = True, vertical_flip = True, fill_mode = 'constant'	0.1269	0.3671	
4	brightness_range = [-1.5, 1.5], shear_range = 0.5, fill_mode = 'nearest', rotation_range = 360, width_shift_range = 0.2, height_shift_range = 0.2	0.2340	0.3307	

Based on the result obtained, the second combination achieved the highest dice coefficient among all the combinations, followed by the first combination. However, by observing the predicted output images of these two combinations, the output generated by the second combination contained more noise than the first combination, as shown in Figure 4.5 below.

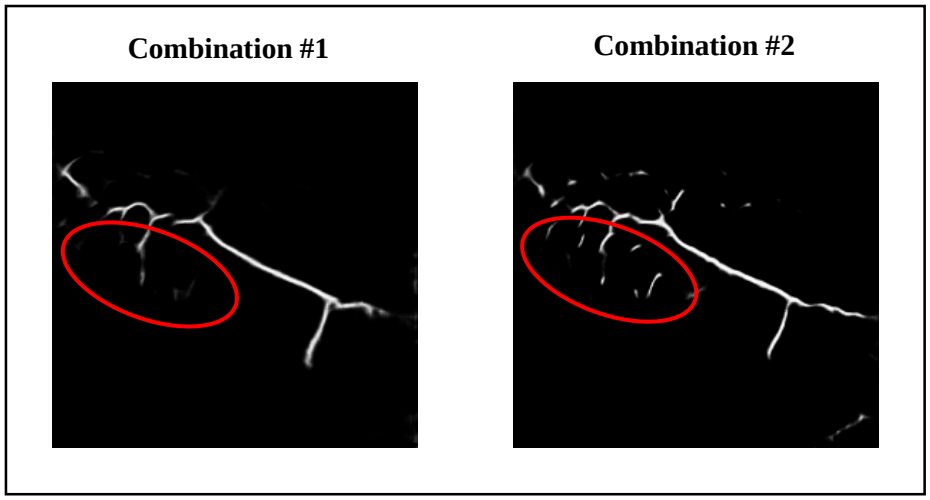


Figure 4.5 Comparison of predicted outputs for combination #1 and #2 in terms of noise.

Besides extra noises being extracted by the first data augmentation combination, it was observed that for some vein images, there were useful pixels either being removed or wiped out due to the extremely strong augmentation strength of the combination. This condition has led to excessive broken connection between the extracted vein line as shown in Figure 4.6. While for the first combination, the broken gaps in vein lines were less severe, resulting in more useful features being preserved.

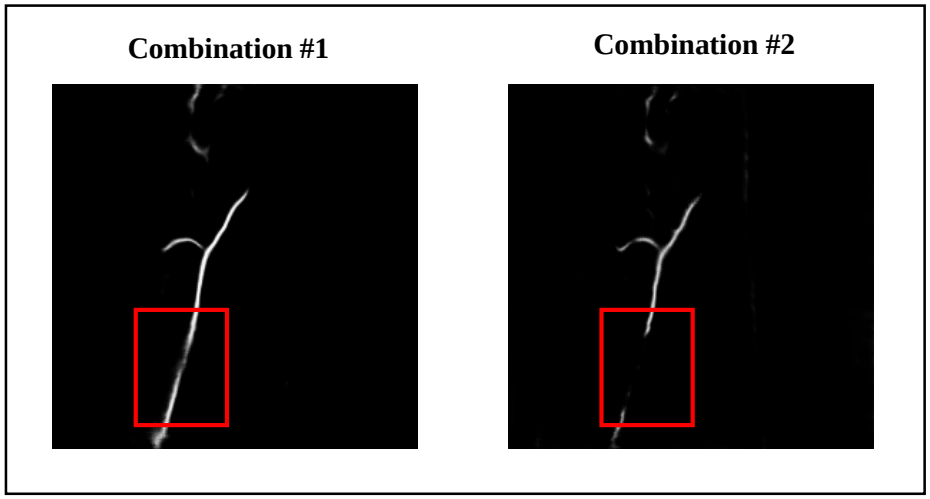


Figure 4.6 Comparison of predicted outputs for combination #1 and #2 in terms of broken connection.

The third and fourth data augmentation combinations were aimed to observe how the vein features can be extracted when a combination of larger zoom range, brightness range, width shift range and height shift range was used.

The third data augmentation technique used a large zoom range, which was [1 – 0.5, 1 + 1.5] to observe its capability to localize the vein. From Figure 4.7 below, it was observed that the extracted vein features in the center of the forearm were clearer with less noise, but large portions of the forearm were being wrongly extracted as veins. This impact was due to the use of a large zoom range that caused the forearm part being discarded or wrongly recognized as useful pixels. Thus, this combination cannot be implemented.

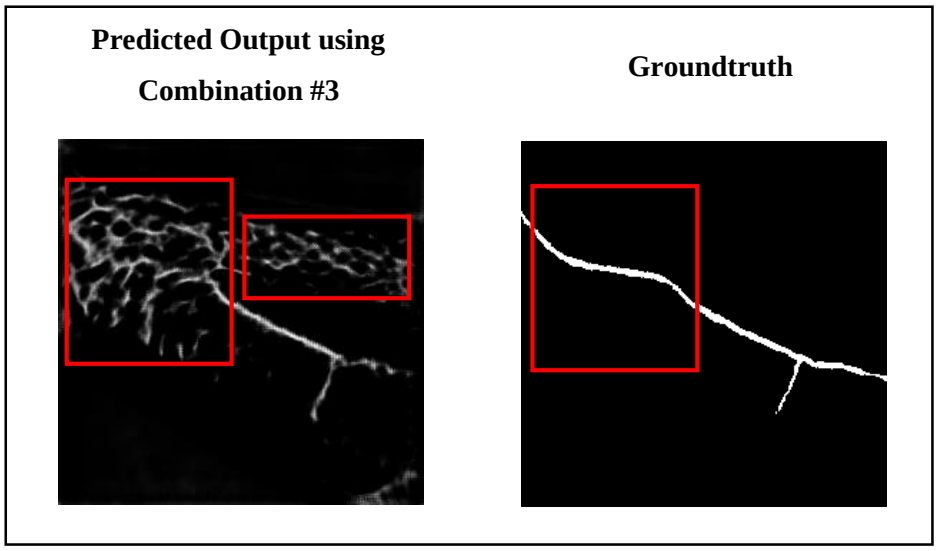


Figure 4.7 Comparison of predicted output and groundtruth using combination #3.

On the other hand, the fourth combination was used to observe the impact of large brightness range and shift ranges on vein feature extraction. The purpose of using large range augmentation was to simulate a situation where the surrounding or lighting was very strong and the forearm position was placed too far from the center. From Figure 4.8, it was observed that almost $\frac{1}{4}$ of the vein features were being removed due to either extremely high brightness or the area was shifted out of range, resulting in zero extraction from the forearm area. The remaining features contained some noises but the vein line was identifiable. Although the predicted output contained less noise compared

to that of the third combination, the missing vein portions can highly affect effective vein localization. Thus, this combination cannot be implemented.

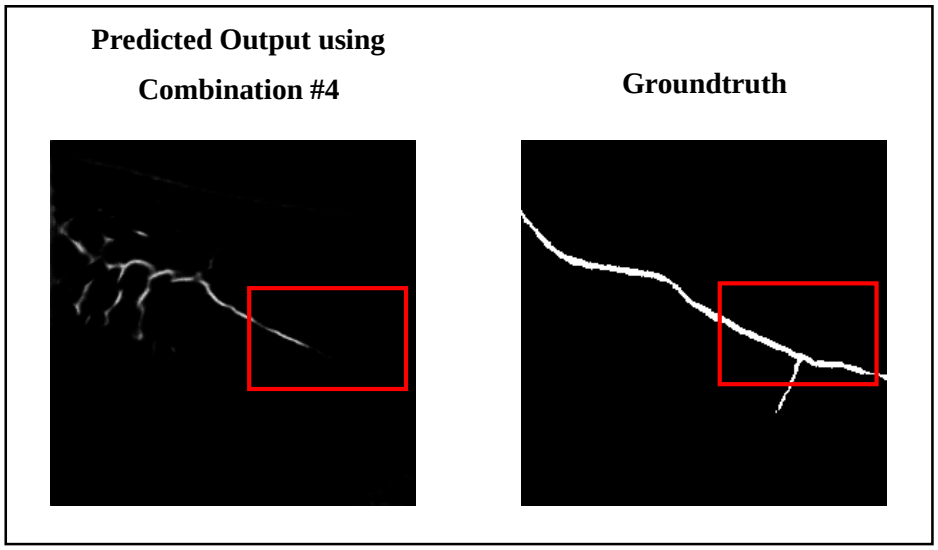


Figure 4.8 Comparison of predicted output and groundtruth using combination #4.

In this section, it was decided that the first combination of data augmentation techniques would be used for subsequent simulations because it had the most satisfied predicted output in terms of dice coefficient and groundtruth with the least presence of noise and loss of vein features.

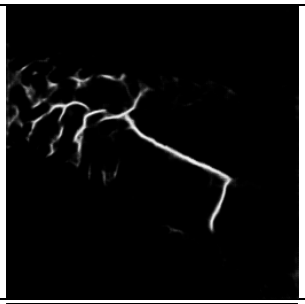
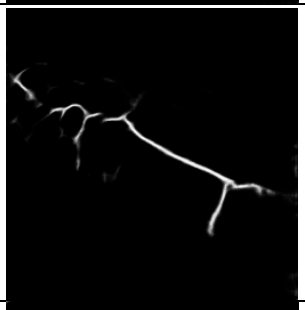
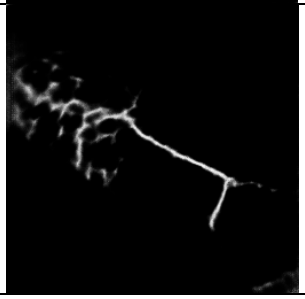
4.4 Analysis of Model Training on Different Hyperparameters

The optimized model, with the application of selected data augmentation combination, was further trained on different hyperparameter settings, which were learning rate (0.001, 0.0001, 0.00001), activation function (ReLU, ELU), and the number of epoch (5, 10, 15, and 25 epochs). The results of the experiments were discussed in the following sections.

4.4.1 Model Training on Different Learning Rates

Three learning rates, which were 0.001, 0.0001, and 0.00001, were chosen to train the model to observe their computed dice coefficients as well as predicted outputs. The training results were shown in Table 4.4 below.

Table 4.4 Dice coefficient scores and predicted outputs for different learning rates.

Learning Rate	Dice Coefficient Scores Obtained		Predicted Output
	Training	Validation	
0.001	0.6246	0.6498	
0.0001	0.6764	0.7032	
0.00001	0.5115	0.5330	

From the results displayed above, it was observed that the model that used a learning rate of 0.0001 achieved the highest dice coefficient score and clearest predicted output among all the learning rates. The model with the largest learning rate (0.001) performed slightly better than the model with the smallest learning rate (0.00001), but both the learning rates had extracted unnecessary forearm features as shown at upper left corners of the sample output images. These extracted noises might increase complexity to localize the veins precisely if the learning rates were chosen to be used.

To elaborate the reason for the observation, the learning rate of 0.001 was considered slightly big for this vein segmentation model. Although the cost has decreased during gradient descent in backpropagation, the big learning rate might cause the algorithm to alternate between two points due to crude update steps. It was difficult to reach the local minimum and thus resulted in a higher loss.

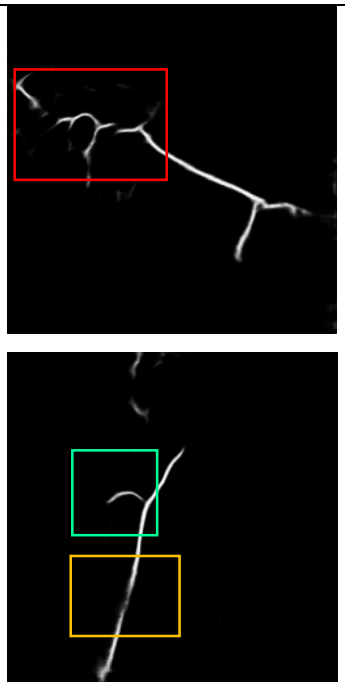
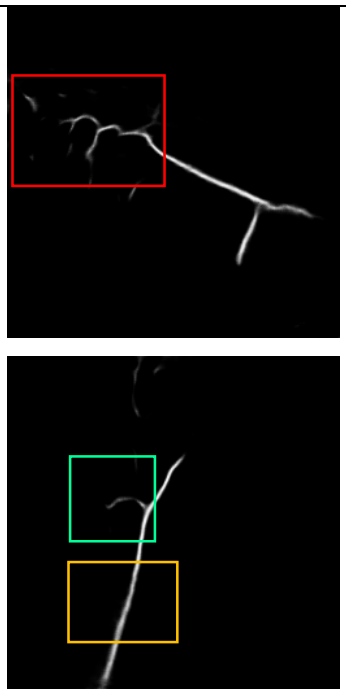
On the other hand, the learning rate of 0.00001 was too small and it was only capable of making small adjustments to the network weights in each step. It will take more time than an appropriate learning rate to reach the local minimum. This might waste training time as shorter training time was preferred to speed up the whole process.

Therefore, the learning rate of 0.0001 was the most suitable to be implemented due to its ability to minimize cost and reach the local minimum within a reasonable number of iterations. Since the learning rate of 0.001 had a higher dice coefficient than learning rate of 0.00001, it was assumed that the optimal learning rate lies between the range of 0.0001 to 0.001. This range will be used for a proper learning rate fine-tuning.

4.4.2 Model Training on Different Activation Functions

After selecting an appropriate learning rate, the model was trained on two different activation functions, which were ReLU and ELU by using 5 epochs. The result obtained was shown in Table 4.5 below.

Table 4.5 Dice coefficient scores and predicted outputs for different activation functions.

Activation Function	Dice Coefficient Scores Obtained		Predicted Output
	Training	Validation	
ReLU	0.6764	0.7032	
ELU	0.6841	0.6529	

Based on the results above, both activations have reflected nearly similar amount of feature information loss from their predicted outputs. For example, ReLU activation showed a greater information loss in the yellow box highlighted on the second sample output, but it was able to extract more vein pixels than ELU activation as seen in the red and green boxes. However, by observing the performance scores, ReLU activation achieved a slightly higher dice coefficient compared to ELU activation.

From a theoretical perspective, ReLU was favoured by most of the deep learning network because it did not saturate in the positive region, resulting in less vanishing gradient effect. However, ReLU was still saturated in the negative region. This might cause a large number of neurons to be inactive if most of the weighted inputs were negative. As a consequence, the activation output was zero and network parameters were unable to be updated.

On the other hand, ELU resolved this issue by having negative saturation regions which enabled neurons to produce negative activation outputs instead of zero activation. Dead ReLU problem can be avoided and the negative activation outputs were able to update network parameters in correct directions.

However, there was a lack of state-of-the-art models that implemented ELU activation in their deep neural network architecture. For example, U-Net architecture developed in [6] and [50] had just followed the existing implementation of ReLU activation in their U-Net models because it was more well-established, and more suitable and common to be used than other activations in the hidden layers.

Since the results of ReLU activation showed a higher dice coefficient and there was not much research that supported the implementation of ELU activation, ReLU activation was more suitable to be applied in this vein extraction U-Net model.

4.4.3 Model Training on Different Number of Epochs

Previous training experiments were conducted using different augmentation combinations, learning rates and activation functions with 5 epochs. This experiment had used 10, 15, and 25 epochs to further reduce dice coefficient loss and to observe underfitting or overfitting in the model. The result obtained was shown in Table 4.6 on the next page.

From the result obtained, it was observed that the vein features learned by the model became more complete and clear when the number of epochs used increased. Figure 4.9 below showed the comparison between the groundtruths and the predicted outputs when the model was trained with 15 and 25 epochs. It can be seen that there was less information loss in 25-epoch predicted output as highlighted in red boxes, indicating that larger epochs can increase the capability of a model to learn more class-specific features from input images.

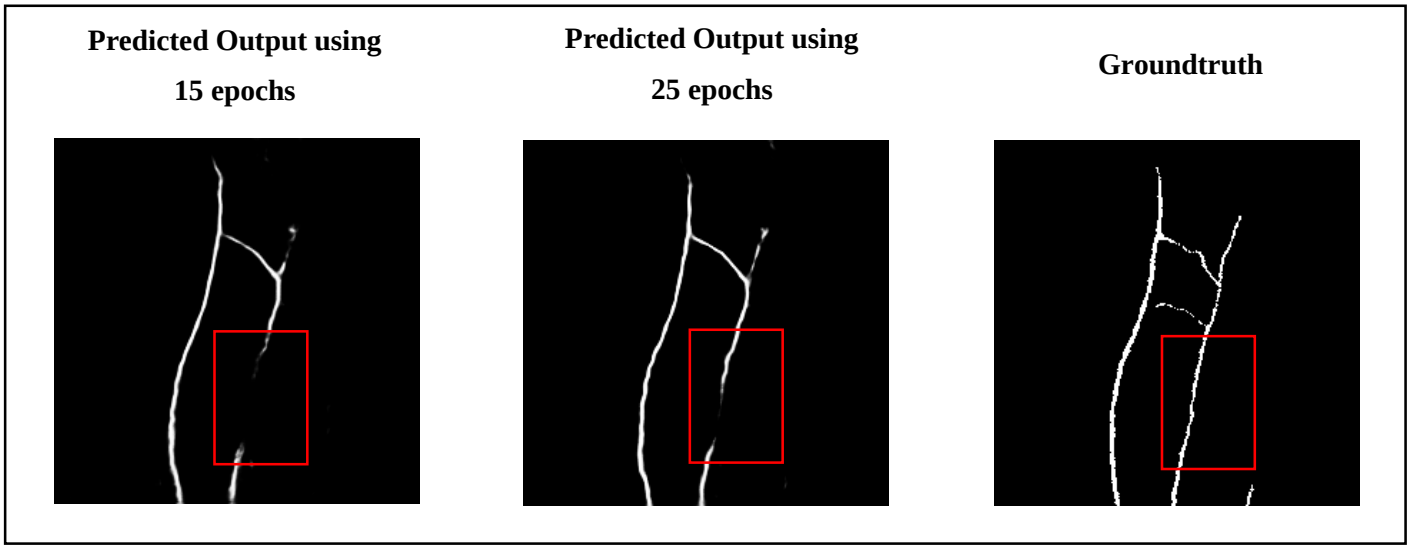
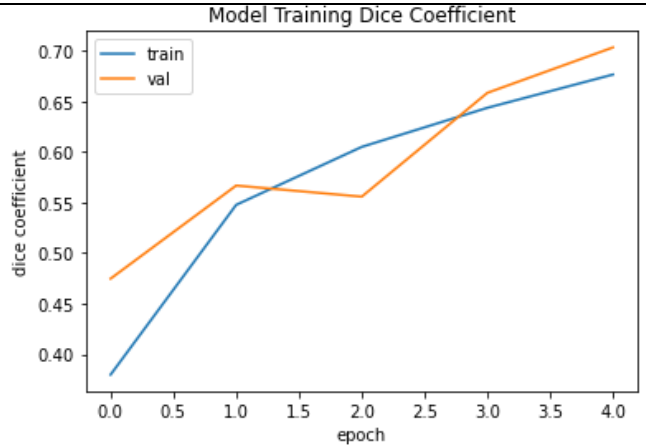
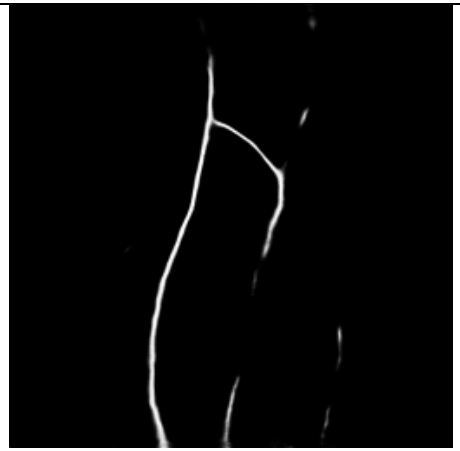
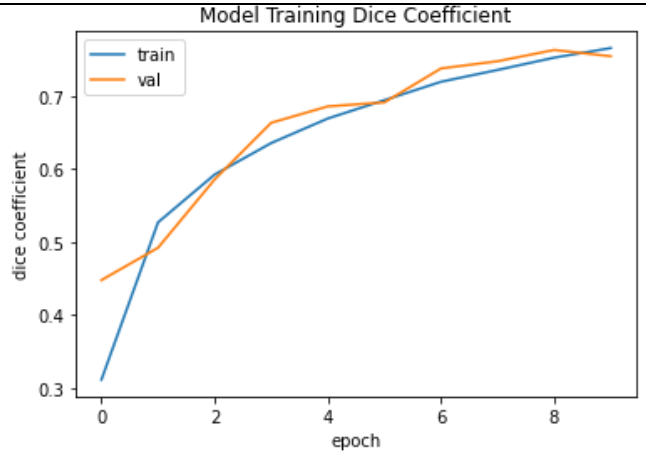
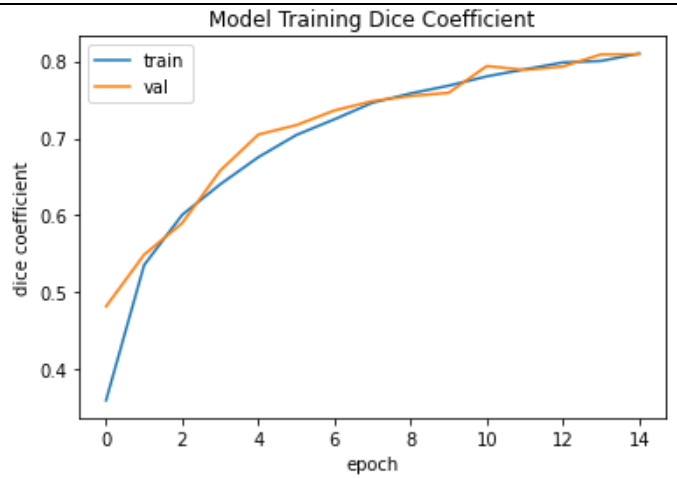
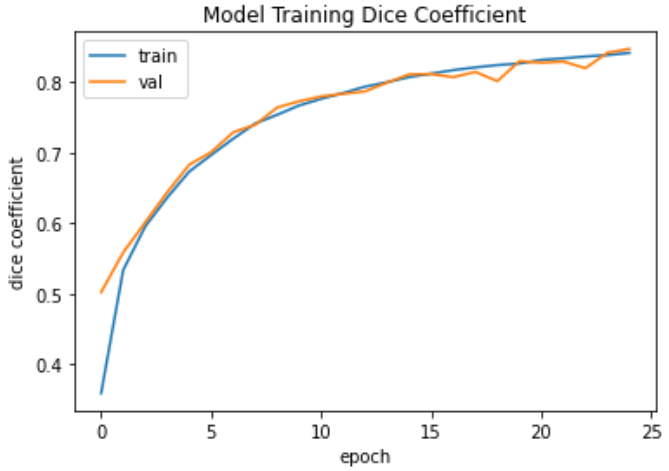


Figure 4.9 Comparison of predicted outputs with groundtruth when different epochs were used.

However, according to [54], a model will face the risk of overfitting when it is trained with too many epochs. From the learning curves shown in Table 4.5, it was assumed that the models had not overfitted the images since the training and validation dice coefficients were close to each other, but this statement has to be proved by predicting on test set images. In this training section, since 25 epochs gave the most satisfied output, it was chosen to be applied in the model.

Table 4.6 Results obtained when different numbers of epochs were used.

Number of Epochs	Dice Coefficient Scores Obtained		Predicted Output	Learning Curve
	Training	Validation		
5	0.6764	0.7032		
10	0.7653	0.7541		

Number of Epochs	Dice Coefficient Scores Obtained		Predicted Output	Learning Curve
	Training	Validation		
15	0.8100	0.8092		
25	0.8420	0.8476		

4.5 Phase 2 Model Training

To reduce the risk of overfitting, the complexity of the network was being counted into consideration as one of the aspects to further optimize the network architecture. By using the first data augmentation combination and the chosen hyperparameter configurations (learning rate = 0.0001, activation function = ReLU, number of epochs = 25), four versions of lightweight U-Net models were built and trained with different number of layer blocks and filter sizes. The dice coefficient scores generated by each model were shown in Table 4.7 below.

Table 4.7 Dice coefficient scores for the optimized lightweight model versions.

Lightweight U-Net Version	Number of Layer Blocks	Filter Size		Dice Coefficient	
		First Convolution Block	Bottleneck Block	Training	Validation
1	8	64	512	0.8338	0.8277
2	6	64	256	0.8304	0.8245
3	10	16	256	0.7962	0.7811
4	8	16	128	0.7408	0.7392

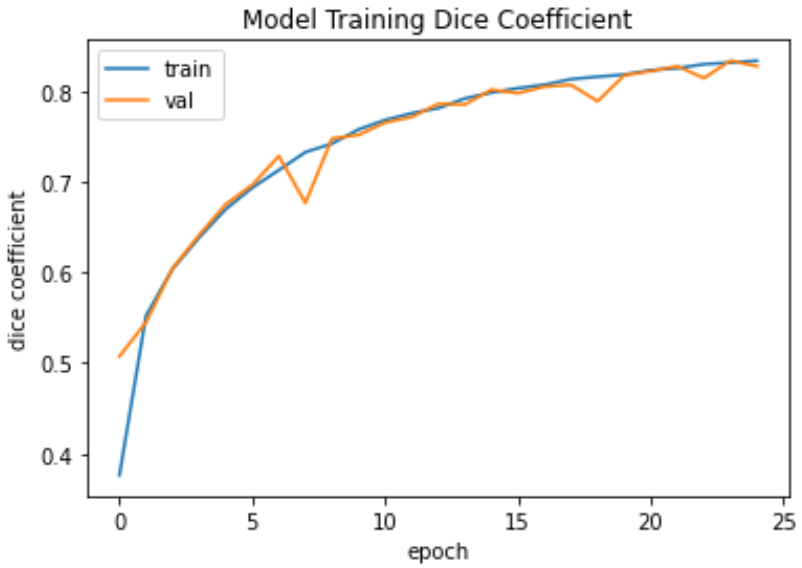


Figure 4.10 Dice coefficient curve for the first lightweight model.

The dice coefficient score obtained by the first lightweight U-Net model was the highest compared to other model versions, which indicated that the number of layer blocks and filter sizes in the blocks were the most suitable to be used in subcutaneous vein segmentation task among all the versions. Thus, the first lightweight model version was chosen for the subsequent hyperparameter fine-tuning. However, the first lightweight model had a slightly lower validation dice coefficient than training dice coefficient. As observed from its dice coefficient curve in Figure 4.10, the dice coefficient fluctuated around the last few epochs, but the distance between training and validation dice coefficient was small. Thus, it can be assumed that the impact of data overfitting was very minimum in this version of lightweight model.

4.6 Hyperparameter Fine-Tuning

Through the intensive training analysis in phase 1 (different optimized architectures, data augmentation combinations and hyperparameters) and phase 2 (different number of layer blocks and filter sizes), the selection details of the finalized model were summarised in Table 4.8 below. The selected configurations were used for the hypermodel in hyperparameter fine-tuning.

Table 4.8 Summarised selection from training analysis.

Model Configuration		**Selection Detail**
Phase 1 Model Training		
Model Architecture		Optimized with Conv2DTranspose and BatchNormalization layers
Data Augmentation Combination		Data Generator #1
Learning Rate		0.0001
Activation Function		ReLU
Number of Epochs		25
Phase 2 Model Training		
Number of Layer Block		8
Filter size	**First Convolution Block**	64
	Bottleneck Block	512

The hyperparameter being fine-tuned in this phase was learning rate. In phase 1 model training, it was determined that the optimal learning rate was lied between the range of 0.0001 to 0.001. Thus, 0.0001 was set as minimum value and 0.001 set as maximum value, so that the tuner would search for the optimum learning rate between this range.

```
kt_learning_rate = kt.Float('steps',
                            min_value = 0.0001,
                            max_value = 0.001,
                            step = 0.00005)
```

Figure 4.11 The configuration of learning rate for hyperparameter fine-tuning.

```
tuner.search(myGene_1,
             steps_per_epoch = 1000,
             epochs = 25,
             validation_data = myGene_1,
             validation_steps = 1000,
             callbacks=[stop_early])

# Get the optimal hyperparameters
best_hps=tuner.get_best_hyperparameters(num_trials=1)[0]
```

```
Trial 17 Complete [00h 07m 56s]
dice_coef: 0.5324798822402954

Best dice_coef So Far: 0.5635988712310791
Total elapsed time: 02h 17m 39s
INFO:tensorflow:Oracle triggered exit
```

Figure 4.12 Result obtained after 17 trials of hyperparameter search.

Figure 4.12 above showed the result of hyperparameter fine-tuning, in which the hyperparameter search was carried out for 17 trials. This result indicated that there were 17 models being trained during the search for the optimum learning rate with the aid of early stopping.

After approximately 2.5 of searching time, the optimum hyperparameters were saved into a variable, which was best_hps. The optimum hyperparameters, which included learning rate, were shown in Figure 4.13 below. It can be observed that the optimum learning rate obtained was 0.00035. This value would be used to retrain the chosen lightweight model.

```
print(best_hps.values)
```

```
{'steps': 0.00035000000000000005, 'tuner/epochs': 2, 'tuner/initial_epoch': 0, 'tuner/bracket': 3, 'tuner/round': 0}
```

Figure 4.13 The optimum hyperparameters obtained.

```
# Build the model with the optimal hyperparameters and train it on the data for 25 epochs
model_tuned = tuner.hypermodel.build(best_hps)
model_checkpoint_tuned = ModelCheckpoint('model_tuned.hdf5', monitor = 'dice_coef',verbose = 1, save_best_only = True)
history_tuned = model_tuned.fit(myGene_1, steps_per_epoch = 1000,epochs = 25, validation_data = myGene_1, validation_steps = 100
```

```
1000/1000 [==============================] - 202s 202ms/step - loss: 0.0045 - accuracy: 0.9980 - dice_coef: 0.8443 - dice_coe
f_loss: 0.1557 - val_loss: 0.0047 - val_accuracy: 0.9979 - val_dice_coef: 0.8404 - val_dice_coef_loss: 0.1596

Epoch 00022: dice_coef did not improve from 0.40468
Epoch 23/25
1000/1000 [==============================] - 202s 202ms/step - loss: 0.0045 - accuracy: 0.9980 - dice_coef: 0.8474 - dice_coe
f_loss: 0.1526 - val_loss: 0.0050 - val_accuracy: 0.9978 - val_dice_coef: 0.8372 - val_dice_coef_loss: 0.1628

Epoch 00023: dice_coef did not improve from 0.40468
Epoch 24/25
1000/1000 [==============================] - 202s 202ms/step - loss: 0.0044 - accuracy: 0.9981 - dice_coef: 0.8500 - dice_coe
f_loss: 0.1500 - val_loss: 0.0050 - val_accuracy: 0.9978 - val_dice_coef: 0.8393 - val_dice_coef_loss: 0.1607

Epoch 00024: dice_coef did not improve from 0.40468
Epoch 25/25
1000/1000 [==============================] - 201s 201ms/step - loss: 0.0043 - accuracy: 0.9981 - dice_coef: 0.8513 - dice_coe
f_loss: 0.1487 - val_loss: 0.0045 - val_accuracy: 0.9980 - val_dice_coef: 0.8470 - val_dice_coef_loss: 0.1530

Epoch 00025: dice_coef did not improve from 0.40468
```

Figure 4.14 Dice coefficient obtained after retraining the hypermodel using best hyperparameter.

Figure 4.14 showed the final dice coefficient scores obtained from hypermodel retraining, which was 0.8513 for training dice coefficient and 0.8470 for validation dice coefficient. Both the dice coefficient scores were higher than the dice coefficients obtained in the original lightweight model (training dice coefficient = 0.8338, validation dice coefficient = 0.8277), indicating that the optimum hyperparameters had indeed improved the hypermodel performance effectively. Figure 4.15 below showed the dice coefficient curve obtained for the retrained hypermodel. The distance between training dice coefficient and validation dice coefficient was close to each other, which proved that there was no data overfitting happening when using the optimum hyperparameters in retraining the hypermodel.

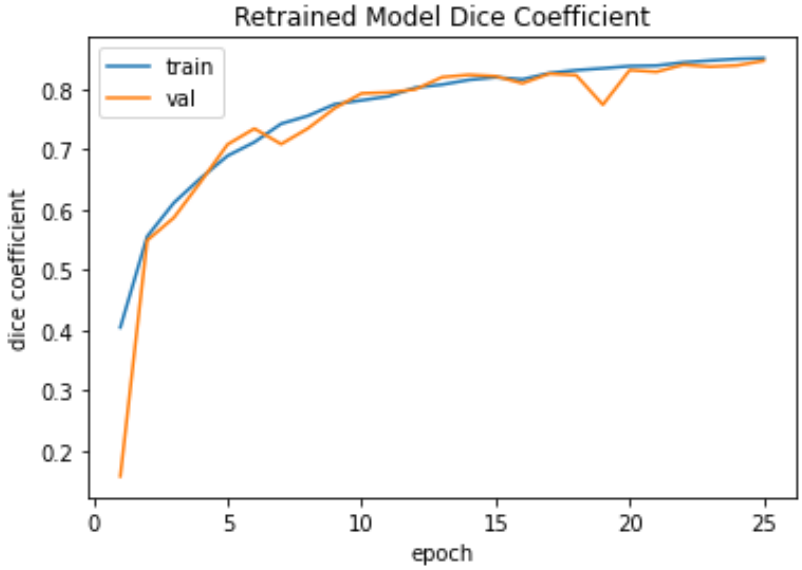


Figure 4.15 Dice coefficient curve for the retrained lightweight model.

4.7 Model Evaluation on Unsupervised Test Set Images

The hypermodel was used to perform prediction on unsupervised test set images, in which the images without groundtruths were used to be segmentated and subsequently generate predicted outputs. Performance metrics used for unsupervised vein segmentation evaluation included confusion matrix, dice coefficient, accuracy, specificity, sensitivity, and precision. Table 4.9 below showed the results obtained by the hypermodel during output prediction.

Table 4.9 Result obtained for model evaluation.

Performance Metric	**Result Obtained**
Confusion Matrix	308 | 2 68 | 242
Dice Coefficient	0.8736
Accuracy	0.8871
Specificity	0.9935
Sensitivity	0.7806
Precision	0.9918

Based on the results shown above, the hypermodel had achieved a higher dice coefficient compared to the dice coefficient scores obtained in model training and retraining phases. The accuracy had slightly dropped by approximately 0.1, indicating that the model might still need to be improved in order to identify vein pixels especially in fork vein branches besides the main vein branch, and also to classify positive and negative vein pixels more accurately.

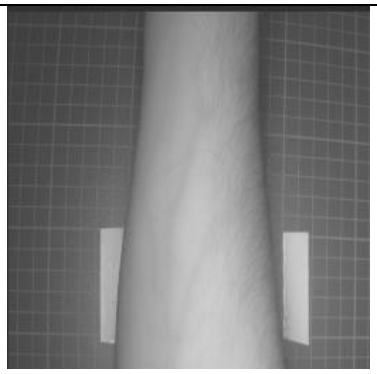
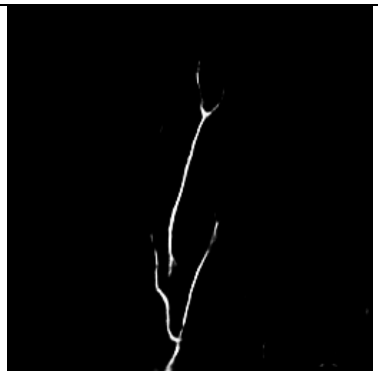
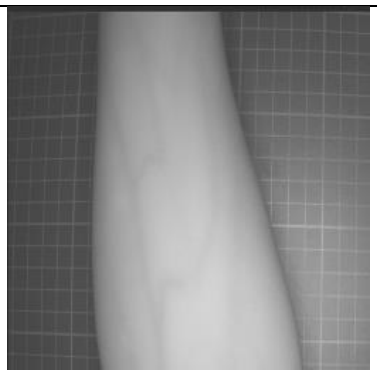
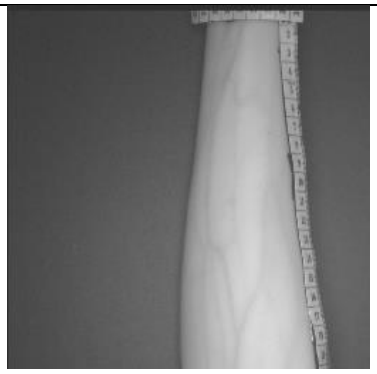
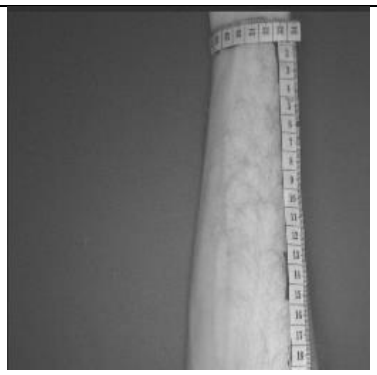
On the other hand, the hypermodel had achieved a relatively high specificity, which meant that the hypermodel was highly capable to predict non-vein pixels correctly. It can be deduced that the model can effectively minimize misclassification of negative pixels, such as patient's forearm or surrounding backgrounds, as vein pixels.

Besides, the sensitivity had improved from 0.4856 from the original U-Net architecture to 0.7806, which could explain that the capability of the lightweight U-Net model in detecting positive vein pixels was greatly enhanced. The high precision also indicated that more than half of the positively predicted pixels were actually true vein pixels. A highly accurate model must have both high recall (sensitivity) and precision [55]. In the experiments, the model had been successfully improved to achieve both high sensitivity and precision by using the lightweight U-Net model.

Table 4.10 showed some examples of model prediction results from four subjects. From the examples shown below, it can be seen that the model was able to extract the main stems and branches of the vein anatomies with a quite satisfying and complete vein feature extraction. In the first three examples, the extracted main stem of the vein, which is the common spot for IV insertion, can be easily differentiated even with bare eyes.

However, there were also some predicted outputs that showed a poor extraction such as the fourth example showed in Table 4.10. This might be due to the existence of noise in the images such as forearm hair, dermis fat, and dark skin tone that had increased the complexity for the model to identify the true vein pixels. As can be seen in the fourth example, the model had extracted some forearm hair pixels as vein pixels, indicating that a post-processing algorithm with length filter would be needed to have a higher specificity in recognizing the true veins from the images. The artefact such as hair can be further filtered by the length and width of the extracted pixels because the true vein has a unique continuous tubular shape as compared to other artefacts' features.

Table 4.10 Examples of vein extraction results.

No.	Forearm Image	Predicted Output
1		
2		
3		
4		

Chapter 5 Conclusion and Future Work

5.1 Conclusion and Objectives Evaluation

IV access is a commonly known challenging clinical procedure for medical staff to localize patients' subcutaneous veins especially for those patients with dark skin tone and thick dermis fat. This scenario often leads to repeated needle insertion that causes the patients to suffer from venepuncture pain. In order to solve and reduce the occurrence of this problem, the motivation of this project was to develop and optimize a deep learning architecture, U-Net, to increase its capability in precise subcutaneous vein localization and segmentation.

This project had improved the existing U-Net architecture by optimizing the original model via transpose convolution and batch normalization. Instead of simply scaling the input activation map through upsampling, transpose convolution has allowed the filters to extract features when reconstructing the activation map back to the original input size. Batch normalization was applied to standardize layer inputs with zero mean and unit standard deviation so that the covariance shift can be decreased.

Besides, intensive hyperparameter analysis was carried out to observe the performance of the optimized U-Net model when different learning rates, activation functions, number of epochs, and complexity of network were applied. Furthermore, this project also explored unsupervised subcutaneous vein segmentation by manually selecting checkpoints in the unsupervised vein images from random directions. This stage consumed the greatest amount of time and effort because selecting an effective and true checkpoint manually was very challenging. After generating predicted outputs, the checkpoints were used to compare the pixel values between the predicted outputs and the unsupervised forearm images.

With these steps being accomplished, the objectives of the project were achieved. Firstly, an optimized deep learning architecture, which was an optimized lightweight U-Net model was developed for a more precise subcutaneous vein extraction and localization. Secondly, the model capability on extracting veins from higher-difficulty forearm images, in which the images contained forearms with thick dermis hair, dark skin tone, and thick dermis fat, was increased greatly. This could be observed through the high accuracy (0.8871), specificity (0.9935), sensitivity (0.7806),

and precision (0.9918) achieved, indicating that the proposed model could extract and localize subcutaneous veins accurately and effectively based on the observation made from the selected checkpoints. Thirdly, unsupervised learning technique was implemented to assess the performance of the proposed model in locating and extracting the true subcutaneous veins. This could be observed from the predicted outputs that showed a satisfying vein extraction result.

Therefore, it could be concluded that the proposed optimized lightweight U-Net model had improved the capability and performance of current venipuncture machines in conducting intravenous procedures and would reduce the error of wrong vein detection. The proposed model could be applied in the venipuncture machines to locate subcutaneous veins for IV procedures.

5.2 Novelties and Contributions

The main contribution of this project was the implementation of unsupervised learning concepts to evaluate model performance. As observed in the literature review and other existing works, most of the vein extraction research, with their purposes for biometric verification and identity authentication, used datasets with groundtruth to carry out experiments. There was a lack of forearm vein datasets available, not to mention the forearm vein datasets with groundtruth, because most of the vein datasets were made up of palm veins or finger veins. To overcome the issue of limited biomedical data, the proposed model was first been trained on a supervised vein dataset and then been evaluated using an unsupervised vein dataset. For each unsupervised forearm images, 20 checkpoints were selected from random directions to evaluate the extraction outcomes of true and false vein pixels by comparing the values of the checkpoints between the unsupervised forearm images and the predicted outputs.

5.3 Future Work

This research project had selected checkpoints manually from the unsupervised forearm images, which was a laborious and time-consuming process in the whole project pipeline. The checkpoints were chosen manually and carried out by human validation. Although the results obtained was satisfied and reliable, the results obtained

might be possible to get biased. In this context, a bigger pool of dataset with the corresponding groundtruth would be needed to further validate the robustness of the proposed optimized lightweight U-Net model in extracting subcutaneous vein accurately. Other than that, improvement will be sought by carrying out more studies to improve the current U-Net architecture for a more sophisticated and prominent deep learning based vein extraction procedure.

REFERENCES

[1] G. B. Beecham and G. Tackling, "Peripheral line placement," in *StatePearls*, Treasure Island, FL, USA: StatPearls, 2021. [Online]. Available: https://www.ncbi.nlm.nih.gov/books/NBK539795/

[2] D. Ahuja, N. Gupta and A. Gupta, "Difficult peripheral intravenous access: need for some light," *Med. J. D.Y. Patil Vidyapeeth*, vol. 13, no. 4, pp. 422 – 423, Jul. 2020, doi: 10.4103/mjdrdypu.mjdrdypu_316_19.

[3] V. A. Yeguas, L. G. Echenique, M. A. T. López, E. C. Domínguez, B. M. Gusmão, E. M. Serrano and M. I. M. Dulanto, "Prevalence of difficult venous access and associated risk factors in highly complex hospitalised patients," *J. Clin. Nurs.*, vol. 26, no. 23 – 24, pp. 4267 – 4275, Feb. 2017, doi: https://doi.org/10.1111/jocn.13750.

[4] R. Das, E. Piciucco, E. Maiorana and P. Campisi, "Convolutional neural network for finger-vein-based biometric identification," *IEEE Trans. Inf. Forensics Secur.*, vol. 14, no. 2, pp. 360 – 373, Feb. 2019, doi: 10.1109/TIFS.2018.2850320.

[5] F. Marattukalam and W. H. Abdulla, "Segmentation of palm vein images using U-Net," presented at the 2020 Asia-Pacific Signal and Information Processing Association Annual Summit and Conference (APSIPA ASC), Auckland, New Zealand, Dec. 7 – 10, 2020, pp. 64 – 70.

[6] P. Wang and H. Qin, "Palm-vein verification based on U-Net," presented at the International Conference on AI and Big Data Application (AIBDA 2019), Guangzhou, China, Dec. 20 – 22, 2019, doi: 10.1088/1757-899x/806/1/012043.

[7] AG Education Ltd. "Intravenous Cannulation (IV)." Medistudents. https://www.medistudents.com/osce-skills/intravenous-cannulation (accessed Aug. 26, 2021).

[8] J. P. Cunha. "What are the Seven Layers of Skin?" eMedicineHealth. https://www.emedicinehealth.com/what_are_the_seven_layers_of_skin/article_em.htm (accessed Aug. 26, 2021).

[9] C. M. Goh, R. Subramaniam, N. M. Saad, S. A. Ali and F. Meriaudeau, "Subcutaneous veins depth measurement using diffuse reflectance images," *Opt.*

Express, vol. 25, no. 21, pp. 25741 – 25759, Oct. 2017, doi: https://doi.org/10.1364/oe.25.025741.

[10] L. T. Tran and H. T. T. Pham, "Designing and building the vein finder system utilizing near-infrared technique," presented at the 7th International Conference on the Development of Biomedical Engineering in Vietnam (BME7), Van Toi, V., Le, T., Ngo, H., Nguyen, TH., Vietnam, Jun. 27 – 29, 2018, vol. 69, pp. 383 – 387, doi: https://doi.org/10.1007/978-981-13-5859-3_68.

[11] F. Alonso. "Neural Networks and Deep Learning. Chapter 1: Prelude." FutureSpace. https://www.futurespace.es/en/redes-neuronales-y-deep-learning-capitulo-1-preludio/ (accessed Aug. 26, 2021).

[12] A. Subasi, "Chapter 2 – Data preprocessing," in *Practical Machine Learning for Data Analysis Using Python*, Cambridge, MA, USA: Academic Press, 2020, pp. 27 – 89. [Online]. Available: https://doi.org/10.1016/B978-0-12-821379-7.00002-3.

[13] A. S. Sadiq, H. Faris, A. M. Al-Zoubi, S. Mirjalili and K. Z. Ghafoor, "Chapter 17 - Fraud detection model based on multi-verse features extraction approach for smart city applications," in *Smart Cities Cybersecurity and Privacy*, Amsterdam, Netherlands: Elsevier, 2019, pp. 241 – 251. [Online]. Available: https://doi.org/10.1016/B978-0-12-815032-0.00017-2.

[14] C. P. Sree, K. S. Naveen, J. A. S. Sampreeth, D. Pavan and V. S. A. Kumar, "Real time object detection with deep learning," *IJAEMA*, vol. 12, no. 5, pp. 1 – 7, May 2020, doi: 18.0002.IJAEMA.2020.V12I5.200001.0156552.

[15] A. Sengur, U. Budak, Y. Akbulut, M. Karabatak and E. Tanyildizi, "7 - A survey on neutrosophic medical image segmentation," in *Neutrosophic Set in Medical Image Analysis*, Cambridge, MA, USA: Academic Press, 2019, pp. 145 – 165, doi: https://doi.org/10.1016/B978-0-12-818148-5.00007-2.

[16] W. Yang, S. Wang, J. Hu, G. Zheng, J. Yang and C. Valli, "Securing deep learning based edge finger vein biometrics with binary decision diagram," *IEEE Trans. Industr. Inform.*, vol. 15, no. 7, pp. 4244 – 4253, Jul. 2019, doi: 10.1109/TII.2019.2900665.

[17] “A Simple Guide to Semantic Segmentation.” BeyondMinds. https://beyondminds.ai/blog/a-simple-guide-to-semantic-segmentation/ (accessed Aug. 26, 2021).

[18] M. Sharma. “A Beginners Guide to Unsupervised Learning.” Medium. https://medium.com/analytics-vidhya/beginners-guide-to-unsupervised-learning-76a575c4e942 (accessed Mar. 23, 2022).

[19] E. W. Ting, M. Z. Ibrahim and D. J. Mulvaney, “Investigation of dimensionality reduction in a finger vein verification system,” in *Lecture Notes in Electrical Engineering*, Springer: Singapore, 2018, vol. 449, pp. 194 – 201. [Online]. Available: https://doi.org/10.1007/978-981-10-6451-7_24.

[20] X. Xi, L. Yang and Y. Yin, “Learning discriminative binary codes for finger vein recognition, *Pattern Recognit.*, vol. 66, pp. 26 – 33, Jun. 2017, doi: https://doi.org/10.1016/j.patcog.2016.11.002.

[21] H. Liu, G. Yang, L. Yang and Y. Yin, “Learning personalized binary codes for finger vein recognition,” *Neurocomputing*, vol. 365, pp. 62 – 70, Nov. 2019, doi: https://doi.org/10.1016/j.neucom.2019.07.057.

[22] H. Qin, X. He, X. Yao and H. Li, “Finger-vein verification based on the curvature in Radon space,” *Expert Syst. Appl.*, vol. 82, pp. 151 – 161, Oct. 2017, doi: https://doi.org/10.1016/j.eswa.2017.03.068.

[23] J. Yang, Y. Shi and G. Jia, “Finger-vein image matching based on adaptive curve transformation,” *Pattern Recognit.*, vol. 66, pp. 34 – 43, Jun. 2017, doi: https://doi.org/10.1016/j.patcog.2017.01.008.

[24] B. Subramaniam and S. Radhakrishnan, “Multiple features and classifiers for vein based biometric recognition,” *Biomed. Res.*, vol. 29, pp. 8 – 13, Feb. 2017, doi: https://doi.org/10.4066/biomedicalresearch.29-16-2318.

[25] N. O’Mahony, S. Campbell, A. Carvalho, S. Harapanahalli, G. V. Hernandez, L. Krpalkova, D. Riordan and J. Walsh, “Deep learning vs. traditional computer vision,” *Adv. Intell. Syst. Comput.*, vol. 943, pp. 128 – 144, Oct. 2019, doi: https://doi.org/10.48550/arXiv.1910.13796.

[26] M. Jogin, Mohana, M. S. Madhulika, G. D. Divya, R. K. Meghana and S. Apoorva, "Feature extraction using convolutional neural networks (CNN) and deep learning," presented at the 3rd IEEE Int. Conf. on RTEICT, Bangalore, India, May 18 – 19, 2018, doi: 10.1109/RTEICT42901.2018.9012507.

[27] J. Phang, "Artificial Intelligence Vs Machine Learning Vs Deep Learning." James Phang. https://www.jamesphang.co.uk/blog/2020/5/17/artificial-intelligence-vs-machine-learning-vs-deep-learning (accessed Aug. 26, 2021).

[28] H. Qin and M. A. El-Yacoubi, "Deep representation-based feature extraction and recovering for finger-vein verification," *IEEE Trans. Inf. Forensics Secur.*, vol. 12, no. 8, pp. 1816 – 1829, Aug. 2017, doi: 10.1109/TIFS.2017.2689724.

[29] H. Qin and M. A. El-Yacoubi, "Deep representation for finger-vein image-quality assessment," *IEEE Trans. Circuits Syst. Video Technol.*, vol. 28, no. 8, pp. 1677 – 1693, Aug. 2018, doi: 10.1109/TCSVT.2017.2684826.

[30] Y. Fang, Q. Wu and W. Kang, "A novel finger vein verification system based on two-stream convolutional network learning," *Neurocomputing*, vol. 290, pp. 100 – 107, May 2018, doi: https://doi.org/10.1016/j.neucom.2018.02.042.

[31] R. Das, E. Piciucco, E. Maiorana and P. Campisi, "Convolutional neural network for finger-vein-based biometric identification," *IEEE Trans. Inf. Forensics Secur.*, vol. 19, no. 2, pp. 360 – 373, Feb. 2019, doi: 10.1109/TIFS.2018.2850320.

[32] I. Boucherit, M. O. Zmirli, H. Hentabli and B. A. Rosdi, "Finger vein identification using deeply-fused convolutional neural network," *Journal of King Saud University – Computer and Information Sciences*, vol. 34, no. 3, pp. 646 – 656, Mar. 2022, doi: https://doi.org/10.1016/j.jksuci.2020.04.002.

[33] E. M. Cherrat, R. Alaoui and H. Bouzahir, "Convolutional neural networks approach for multimodal biometric identification system using the fusion of fingerprint, finger-vein and face images," *PeerJ Comput. Sci.*, vol. 6, no. 248, Jan. 2020, doi: https://doi.org/10.7717/peerj-cs.248.

[34] R. S. Kuzu, E. Piciucco, E. Maiorana and P. Campisi, "On-the-fly finger-vein-based biometric recognition using deep neural networks," *IEEE Trans. Inf. Forensics Secur.*, vol. 15, pp. 2641 – 2654, Feb. 2020, doi: 10.1109/TIFS.2020.2971144.

[35] J. M. Song, W. Kim and K. R. Park, “Finger-vein recognition based on deep DenseNet using composite image,” *IEEE Access*, vol. 7, pp. 66845 – 66863, May 2019, doi: 10.1109/ACCESS.2019.2918503.

[36] R. S. Kuzu, E. Maiorana and P. Campisi, “Vein-based biometric verification using transfer learning,” presented at the 2020 43rd International Conference on Telecommunications and Signal Processing (TSP), Milan, Italy, Jul. 7 – 9, 2020, doi: 10.1109/TSP49548.2020.9163491.

[37] K. J. Noh, J. Choi, J. S. Hong and K. R. Park, “Finger-vein recognition based on densely connected convolutional network using score-level fusion with shape and texture images,” *IEEE Access*, vol. 8, pp. 96748 – 96766, May 2020, doi: 10.1109/ACCESS.2020.2996646.

[38] W. Liu, W. Li, L. Sun, L. Zhang and P. Chen, “Finger vein recognition based on deep learning,” presented at the 2017 12th IEEE Conference on Industrial Electronics and Applications (ICIEA), Siem Reap, Cambodia, Jun. 18 – 20, 2017, pp. 205 – 210, doi: 10.1109/ICIEA.2017.8282842.

[39] G. Meng, P. Fang and B. Zhang, “Finger vein recognition based on convolutional neural network,” presented at the 2017 International Conference on Electronic Information Technology and Computer Engineering (EITCE 2017), Zhuhai, China, Sep. 23 – 24, 2017, vol. 128, doi: https://doi.org/10.1051/matecconf/201712804015.

[40] R. Raghavendra, S. Venkatesh, K. B. Raja and C. Busch, “Transferable deep convolutional neural network features for finger vein presentation attack detection,” presented at the 2017 5th International Workshop on Biometrics and Forensics (IWBF), Conventry, UK, Apr. 4 – 5, 2017, pp. 1 – 5, doi: 10.1109/IWBF.2017.7935108.

[41] S. Fairuz, M. H. Habaebi and E. M. A. Elsheikh, “Finger vein identification based on transfer learning of AlexNet,” presented at the 2018 7th International Conference on Computer and Communication Engineering (ICCCE), Kuala Lumpur, Malaysia, Sep. 19 – 20, 2018, pp. 465 – 469, doi: 10.1109/ICCCE.2018.8539256.

[42] H. G. Hong, M. B. Lee and K. R. Park, “Convolutional neural network-based finger-vein recognition using NIR image sensors,” *Sensors (Basel)*, vol. 17, no. 6, pp. 1297, Jun. 2017, doi: https://doi.org/10.3390/s17061297.

[43] N. A. Al-Johania and L. A. Elrefaei, “Dorsal hand vein recognition by convolutional neural networks: feature learning and transfer learning approaches,” *Int. J. Intell. Eng. Syst.*, vol. 12, no. 3, pp. 178 – 191, Jan. 2019, doi: https://doi.org/10.22266/ijies2019.0630.19.

[44] G. S. Harshan, S. Rayudu, K. A. Bharadwaj and K. Saravanan, “Finger vein recognition using VGG-16 CNN algorithm,” *IJITEE,* vol. 9, no. 8, pp. 112 – 114, May 2020, doi: https://doi.org/10.35940/ijitee.H6254.069820.

[45] M. Mohaghegh and A. Payne, “Automated biometric identification using dorsal hand images and convolutional neural networks,” *J. Phys.: Conf. Ser.*, vol. 1880, no. 1, pp. 12 – 14, Feb. 2021, doi: https://doi.org/10.1088/1742-6596/1880/1/012014.

[46] C. Tang, S. Xia, M. Qian and B. Wang, “Deep learning-based vein localization on embedded system,” *IEEE Access*, vol. 9, pp. 27916 – 27927, Feb. 2021, doi: 10.1109/ACCESS.2021.3058014.

[47] Z. Tao, X. Zhou, Z. Xu, S. Lin, Y. Hu and T. Wei, “Finger-vein recognition using bidirectional feature extraction and transfer learning,” *Math. Probl. Eng.*, May 2021, doi: http://doi.org/10.1155/2021/6664809.

[48] L. Zhang, Z. Cheng, Y. Shen and D. Wang, “Palmprint and palmvein recognition based on DCNN and a new large-scale contactless palmvein dataset,” *Symmetry*, vol. 10, no. 4, Mar. 2018, doi: https://doi.org/10.3390/sym10040078.

[49] E. Jalilian and A. Uhl, “Enhanced segmentation-CNN based finger-vein recognition by joint training with automatically generated and manual labels,” presented at the 2019 IEEE 5th International Conference on Identity, Security, and Behavior Analysis (ISBA), Hyderabad, India, Jan. 22 – 24, 2019, pp. 1 – 8, doi: 10.1109/ISBA.2019.8778522.

[50] J. Zeng, F. Wang, J. Deng, C. Qin, Y. Zhai, J. Gan and V. Piuri, “Finger vein verification algorithm based on fully convolutional neural network and conditional random field,” *IEEE Access*, vol. 8, pp. 65402 – 65419, Apr. 2020, doi: 10.1109/ACCESS.2020.2984711.

[51] X. J. Chan, C. M. Goh, M. T. Chai, A. Z. S. A. Sayed and P. V. Wong, “An automatic vein detection system using deep learning for intravenous (IV) access

procedure," presented at the 2021 IEEE National Biomedical Engineering Conference, Kuala Lumpur, Malaysia, Nov. 9 – 10, 2021, pp. 106 – 111, doi: 10.1109/NBEC53282.2021.9618752.

[52] O. Ronneberger, P. Fischer and T. Brox, "U-Net: convolutional networks for biomedical image segmentation," in *Medical Image Computing and Computer-Assisted Intervention – MICCAI 2015*, New York: Springer, 2015, pp. 234 – 241. [Online]. Available: https://doi.org/10.1007/978-3-319-24574-4_28

[53] M. R. NarasingaRao, V. V. Prasad, P. S. Teja, M. Zindavali and O. P. Reddy, "A survey on prevention of overfitting in convolution neural networks using machine learning techniques," *Int. J. Eng. Technol.*, vol. 7, no. 2.32, pp. 177 – 180, May 2018, doi: 10.14419/ijet.v7i2.32.15399.

[54] S. Afaq and S. Rao, "Significance of epochs on training a neural network," *IJSTR*, vol. 9, no. 6, pp. 485 – 488, Jun. 2020.

[55] J. Weaver, B. Moore, A. Reith, J. McKee and D. Lunga, "A comparison of machine learning techniques to extract human settlements from high resolution imagery," presented at the IGARSS 2018 - 2018 IEEE International Geoscience and Remote Sensing Symposium, Valencia, Spain, Jul. 22 – 27, 2018, pp. 6412 – 6415, doi: 10.1109/IGARSS.2018.8518528.

APPENDIX A

A.1 Weekly Report

FINAL YEAR PROJECT WEEKLY REPORT

(Project II)

Trimester, Year: Trimester 3 Year 3	**Study week no.:** 1
Student Name & ID: Chan Xiao Jing, 18ACB04918	
Supervisor: Dr. Goh Chuan Meng	
Project Title: An Optimal Near-Infrared Subcutaneous Vein Extraction Using Deep Learning	

1. WORK DONE

[Please write the details of the work done in the last fortnight.]

- Reviewed the system codes
- Determined the aspects to be improved

2. WORK TO BE DONE

- Decide the aspects to be improved
- Start to improve the coding part

3. PROBLEMS ENCOUNTERED

- Had difficulties in determine the suitable parts to be improved although the project is not perfect yet

4. SELF EVALUATION OF THE PROGRESS

- Ideas are not novel enough to gain enough marks in FYP2

Supervisor's signature

Student's signature

FINAL YEAR PROJECT WEEKLY REPORT

(Project II)

Trimester, Year: Trimester 3 Year 3	**Study week no.:** 2
Student Name & ID: Chan Xiao Jing, 18ACB04918	
Supervisor: Dr. Goh Chuan Meng	
Project Title: An Optimal Near-Infrared Subcutaneous Vein Extraction Using Deep Learning	

1. WORK DONE

[Please write the details of the work done in the last fortnight.]

- Read articles and journals about subcutaneous vein extraction that implemented deep learning techniques
- Researched on available codes in GitHub
- Started to work on project codes

2. WORK TO BE DONE

- Study more research papers and journals
- Further improve the current model architecture

3. PROBLEMS ENCOUNTERED

- Had difficulties understanding some of the journals
- Had difficulties implementing the ideas found

4. SELF EVALUATION OF THE PROGRESS

- Need to put more effort in understanding the journals and source codes
- Need to speed up the progress of optimizing model architecture

meng

Supervisor's signature

Chan

Student's signature

FINAL YEAR PROJECT WEEKLY REPORT

(Project II)

Trimester, Year: Trimester 3 Year 3	**Study week no.:** 3
Student Name & ID: Chan Xiao Jing, 18ACB04918	
Supervisor: Dr. Goh Chuan Meng	
Project Title: An Optimal Near-Infrared Subcutaneous Vein Extraction Using Deep Learning	

1. WORK DONE

[Please write the details of the work done in the last fortnight.]

- Read articles and journals about subcutaneous vein extraction that implemented deep learning techniques
- Decided the aspects to be improved
- Worked on the coding of the improved system

2. WORK TO BE DONE

- Complete 50% of the coding part
- Get unsupervised dataset and determine a suitable method to work on it

3. PROBLEMS ENCOUNTERED

- Had difficulties implementing the coding
- Encountered programming errors

4. SELF EVALUATION OF THE PROGRESS

- Need to spend more time on optimize the codes of the system
- Study more about unsupervised image segmentation

Supervisor's signature

Student's signature

FINAL YEAR PROJECT WEEKLY REPORT

(Project II)

Trimester, Year: Trimester 3 Year 3	**Study week no.:** 4
Student Name & ID: Chan Xiao Jing, 18ACB04918	
Supervisor: Dr. Goh Chuan Meng	
Project Title: An Optimal Near-Infrared Subcutaneous Vein Extraction Using Deep Learning	

1. WORK DONE

[Please write the details of the work done in the last fortnight.]

- Completed 50% of the coding part
- Read articles and journals about subcutaneous vein extraction that implemented deep learning techniques

2. WORK TO BE DONE

- Complete 80% of the coding part
- Finalize the aspects to be improved

3. PROBLEMS ENCOUNTERED

- Had difficulties implementing the coding
- Encountered programming errors, especially in the part of locating coordinates of true vein pixels

4. SELF EVALUATION OF THE PROGRESS

- Need to spend more time on optimize the codes of the system
- Study more about unsupervised image segmentation

Supervisor's signature

Student's signature

FINAL YEAR PROJECT WEEKLY REPORT

(Project II)

Trimester, Year: Trimester 3 Year 3	**Study week no.:** 5
Student Name & ID: Chan Xiao Jing, 18ACB04918	
Supervisor: Dr. Goh Chuan Meng	
Project Title: An Optimal Near-Infrared Subcutaneous Vein Extraction Using Deep Learning	

1. WORK DONE

[Please write the details of the work done in the last fortnight.]

- Completed 80% of the coding part
- Read articles and journals about subcutaneous vein extraction that implemented deep learning techniques

2. WORK TO BE DONE

- Complete 100% of the coding part
- Start to work on Chapter 1 Introduction (do minor changes according to the current solution)
- Review the system codes

3. PROBLEMS ENCOUNTERED

- Had difficulties implementing the coding
- Encountered programming errors, especially in the part of locating coordinates of true vein pixels

4. SELF EVALUATION OF THE PROGRESS

- Need to devote more time to do the project
- Need to carefully review the codes

Supervisor's signature

Student's signature

FINAL YEAR PROJECT WEEKLY REPORT

(Project II)

Trimester, Year: Trimester 3 Year 3	**Study week no.:** 6
Student Name & ID: Chan Xiao Jing, 18ACB04918	
Supervisor: Dr. Goh Chuan Meng	
Project Title: An Optimal Near-Infrared Subcutaneous Vein Extraction Using Deep Learning	

1. WORK DONE

[Please write the details of the work done in the last fortnight.]

- Complete 100% of the coding part
- Complete minor change on Chapter 1 Introduction according to the current optimized solution
- Reviewed the system codes (need to be done every week)

2. WORK TO BE DONE

- Review the system codes
- Complete checking and minor modification on Chapter 2

3. PROBLEMS ENCOUNTERED

- Felt that the system was still not perfect
- Cannot find additional method that is capable to be done to improve model architecture

4. SELF EVALUATION OF THE PROGRESS

- Need to read more journals and articles related to subcutaneous vein segmentation using U-Net as the baseline model
- Need to devote more time to do the project

Supervisor's signature

Student's signature

FINAL YEAR PROJECT WEEKLY REPORT

(Project II)

Trimester, Year: Trimester 3 Year 3	**Study week no.:** 7
Student Name & ID: Chan Xiao Jing, 18ACB04918	
Supervisor: Dr. Goh Chuan Meng	
Project Title: An Optimal Near-Infrared Subcutaneous Vein Extraction Using Deep Learning	

1. WORK DONE

[Please write the details of the work done in the last fortnight.]

- Complete 20% of Chapter 3
- Complete checking and minor modification on Chapter 2
- Reviewed the system codes of fine-tuning

2. WORK TO BE DONE

- Complete 50% of Chapter 3
- Review the system codes

3. PROBLEMS ENCOUNTERED

- Felt that the model architecture was still not perfect
- Encountered errors when refining the system codes

4. SELF EVALUATION OF THE PROGRESS

- Need to understand more available source codes about vein segmentation using deep learning
- Need to devote more time to do the project

Supervisor's signature

Student's signature

FINAL YEAR PROJECT WEEKLY REPORT

(Project II)

Trimester, Year: Trimester 3 Year 3	**Study week no.:** 8
Student Name & ID: Chan Xiao Jing, 18ACB04918	
Supervisor: Dr. Goh Chuan Meng	
Project Title: An Optimal Near-Infrared Subcutaneous Vein Extraction Using Deep Learning	

1. WORK DONE

[Please write the details of the work done in the last fortnight.]

- Retrained the model and obtained a new result after refining the codes
- Complete 50% of Chapter 3

2. WORK TO BE DONE

- Review the system codes
- Complete Chapter 3

3. PROBLEMS ENCOUNTERED

- Encountered some errors when doing the codes
- Need to spend time on other projects due to deadlines

4. SELF EVALUATION OF THE PROGRESS

- Need to put more effort in Python coding
- Need to devote more time to do the project

meng

Supervisor's signature

Chan

Student's signature

FINAL YEAR PROJECT WEEKLY REPORT

(Project II)

Trimester, Year: Trimester 3 Year 3	**Study week no.:** 9
Student Name & ID: Chan Xiao Jing, 18ACB04918	
Supervisor: Dr. Goh Chuan Meng	
Project Title: An Optimal Near-Infrared Subcutaneous Vein Extraction Using Deep Learning	

1. WORK DONE

[Please write the details of the work done in the last fortnight.]

- Complete Chapter 3
- Complete 30% of Chapter 4
- Read some journals regarding other model architectures for vein segmentation

2. WORK TO BE DONE

- Complete Chapter 4
- Do experiments on other model architectures for possible refinements

3. PROBLEMS ENCOUNTERED

- Encountered some errors when writing the code
- Difficult to fully understand the model architectures in existing studies

4. SELF EVALUATION OF THE PROGRESS

- Need have better time management to do the project and other assignments
- Need to speed up the progress in report writing since next week is draft submission

meng

Supervisor's signature

Chan

Student's signature

FINAL YEAR PROJECT WEEKLY REPORT

(Project II)

Trimester, Year: Trimester 3 Year 3	**Study week no.:** 10
Student Name & ID: Chan Xiao Jing, 18ACB04918	
Supervisor: Dr. Goh Chuan Meng	
Project Title: An Optimal Near-Infrared Subcutaneous Vein Extraction Using Deep Learning	

1. WORK DONE

[Please write the details of the work done in the last fortnight.]

- Complete 40% Chapter 4
- Reviewed the system codes
- Submitted report draft for review

2. WORK TO BE DONE

- Complete Chapter 4
- Check and finalize the system codes

3. PROBLEMS ENCOUNTERED

- Insufficient time to complete the report draft
- Unable to code additional useful system architectures

4. SELF EVALUATION OF THE PROGRESS

- Need to improve self-ability to accomplish tasks nicely in a packed time period
- Need to enhance programming skills in deep learning

Supervisor's signature

Student's signature

FINAL YEAR PROJECT WEEKLY REPORT

(Project II)

Trimester, Year: Trimester 3 Year 3	**Study week no.:** 11
Student Name & ID: Chan Xiao Jing, 18ACB04918	
Supervisor: Dr. Goh Chuan Meng	
Project Title: An Optimal Near-Infrared Subcutaneous Vein Extraction Using Deep Learning	

1. WORK DONE

[Please write the details of the work done in the last fortnight.]

- Complete Chapter 4
- Finalized coding implementation

2. WORK TO BE DONE

- Complete Chapter 5
- Design poster

3. PROBLEMS ENCOUNTERED

- Unexpected error in the coding implementation (already resolved)

4. SELF EVALUATION OF THE PROGRESS

- Need to improve self-ability to accomplish tasks nicely in a packed time period
- Need to enhance programming skills in deep learning

Supervisor's signature

Student's signature

FINAL YEAR PROJECT WEEKLY REPORT

(Project II)

Trimester, Year: Trimester 3 Year 3	**Study week no.:** 12
Student Name & ID: Chan Xiao Jing, 18ACB04918	
Supervisor: Dr. Goh Chuan Meng	
Project Title: An Optimal Near-Infrared Subcutaneous Vein Extraction Using Deep Learning	

1. WORK DONE

[Please write the details of the work done in the last fortnight.]

- Completed Chapter 5
- Complete poster design

2. WORK TO BE DONE

- Complete and check the whole report
- Check the system codes

3. PROBLEMS ENCOUNTERED

- Insufficient time to complete report checking due to deadlines of other assignments

4. SELF EVALUATION OF THE PROGRESS

- Need to improve self-ability to accomplish tasks nicely in a packed time period
- Need to enhance programming skills in deep learning

Supervisor's signature

Student's signature

FINAL YEAR PROJECT WEEKLY REPORT

(Project II)

Trimester, Year: Trimester 3 Year 3	**Study week no.:** 13
Student Name & ID: Chan Xiao Jing, 18ACB04918	
Supervisor: Dr. Goh Chuan Meng	
Project Title: An Optimal Near-Infrared Subcutaneous Vein Extraction Using Deep Learning	

1. WORK DONE

[Please write the details of the work done in the last fortnight.]

- Further enhanced the report and done report checking
- Signed necessary documents
- Submitted the report

2. WORK TO BE DONE

- Do presentation slides
- Identify key points to be put inside the slides

3. PROBLEMS ENCOUNTERED

- Inadequate time to complete report checking within target date (already resolved)

4. SELF EVALUATION OF THE PROGRESS

- Need to enhance programming skills in deep learning
- Need to enhance understanding in subcutaneous vein segmentation from other researches

Supervisor's signature

Student's signature

A.2 Poster

UNIVERSITI TUNKU ABDUL RAHMAN
Faculty of Information and Communication Technology (FICT)

AN OPTIMAL NEAR-INFRARED SUBCUTANEOUS VEIN EXTRACTION USING DEEP LEARNING

Done By: Chan Xiao Jing (18ACB04918)

Supervisor: Dr. Goh Chuan Meng **Moderator: Dr. Tan Hung Khoon**

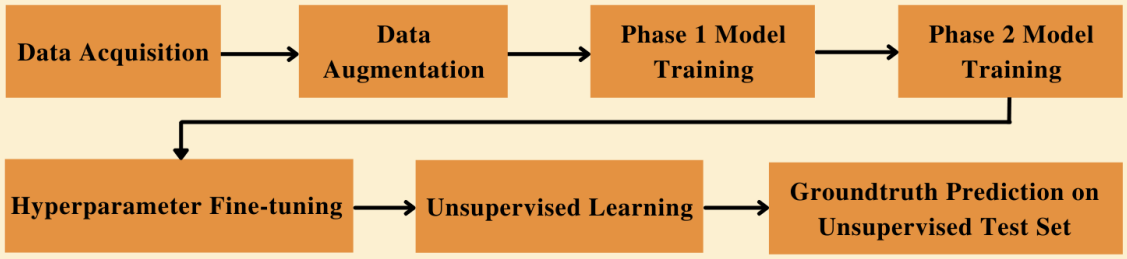
INTRODUCTION

Intravenous (IV) access is a challenging procedure where **clinicians are suffering in locating the subcutaneous vein** due to physiological factors such as hairy forearm, dark skin tone, and thick dermis fat. This has **caused the patients to suffer from venepuncture pain** due to multiple attempts of IV insertion. Hence, **an optimized lightweight U-Net model** was proposed for precise subcutaneous vein localization and extraction using deep learning.

OBJECTIVES

- To develop an optimised deep learning architecture for a more precise subcutaneous vein extraction and localization
- To assess model capability on extracting vein features from higher-difficulty forearm images
- To assess the performance of subcutaneous vein segmentation using unsupervised learning technique

METHODOLOGY



RESULT AND DISCUSSION

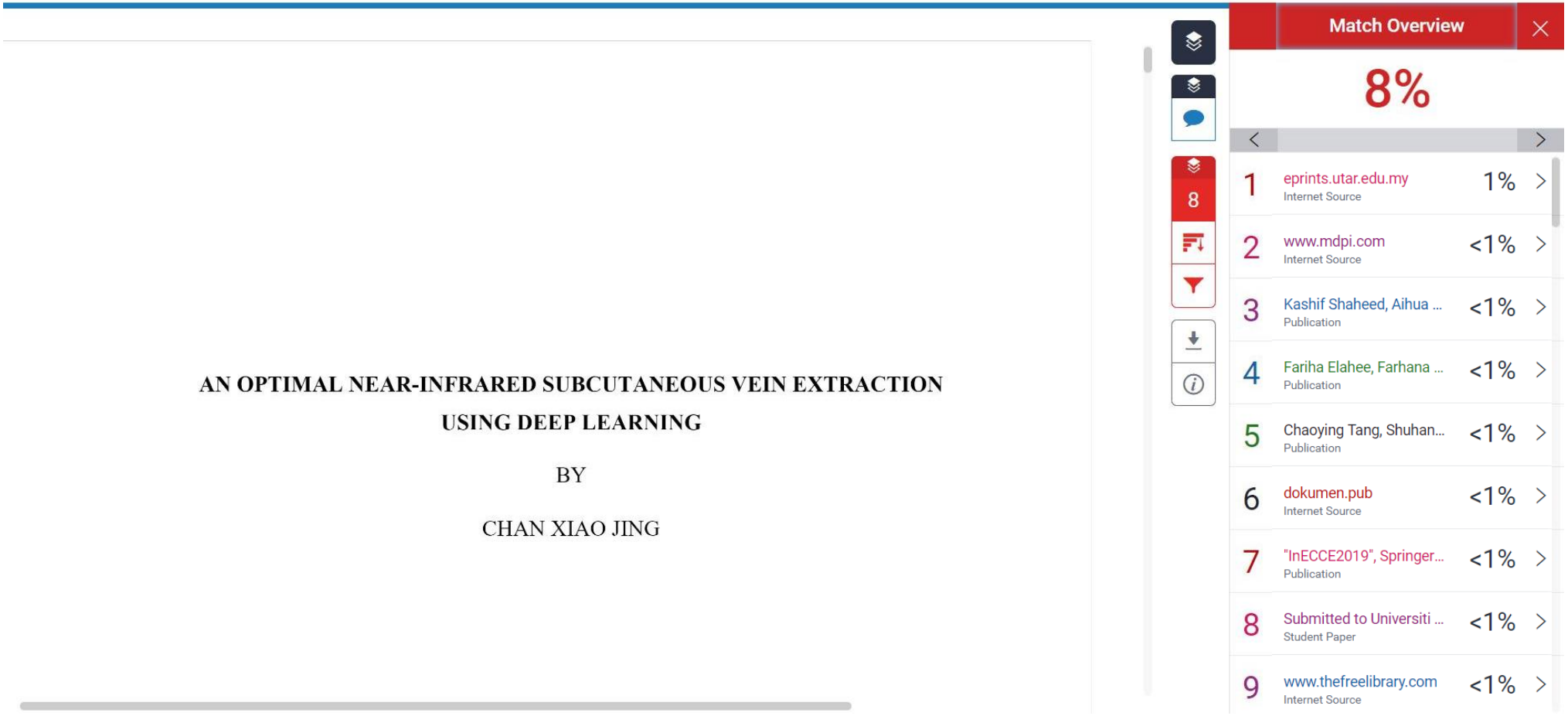
Accuracy : 0.8871 **Specificity : 0.9935**
Sensitivity : 0.7806 **Precision : 0.9918**

- High specificity showed that the model could predict and classify non-vein pixels correctly and effectively
- Improved sensitivity explained the great enhancement of model capability in detecting positive vein pixels
- High precision indicated that more than half of the positively predicted pixels were true vein pixels
- Accuracy has dropped slightly, indicating that further model improvement was needed to identify vein pixels in both the main and fork vein branches

CONCLUSION

- Improved the existing U-Net architecture by **optimizing the original model** via transpose convolution and batch normalization
- **Intensive hyperparameter analysis** was carried out to observe model performance in different learning rates, activation functions, number of epochs, and network complexity
- Applied **unsupervised learning** by manually selecting checkpoints in the unsupervised vein images from random directions
- **Achieved all the project objectives**, showing that the proposed model can be applied in venipuncture machines for IV procedures practically

PLAGIARISM CHECK RESULT



FYP2 report

ORIGINALITY REPORT

8%	5%	6%	2%
SIMILARITY INDEX	INTERNET SOURCES	PUBLICATIONS	STUDENT PAPERS

PRIMARY SOURCES

1	eprints.utar.edu.my Internet Source	1%
2	www.mdpi.com Internet Source	<1%
3	www.rcciit.org Internet Source	<1%
4	Submitted to Universiti Tunku Abdul Rahman Student Paper	<1%
5	Kashif Shaheed, Aihua Mao, Imran Qureshi, Munish Kumar, Sumaira Hussain, Xingming Zhang. "Recent advancements in finger vein recognition technology: Methodology, challenges and opportunities", Information Fusion, 2022 Publication	<1%
6	Fariha Elahee, Farhana Mim, Faizah Binte Naquib, Sharika Tabassom, Tonmoy Hossain, Kazi A Kalpoma. "Comparative Study of Deep Learning based Finger Vein Biometric Authentication Systems", 2020 2nd	<1%

PLAGIARISM CHECK RESULT

Universiti Tunku Abdul Rahman			
Form Title: Supervisor's Comments on Originality Report Generated by Turnitin for Submission of Final Year Project Report (for Undergraduate Programmes)			
Form Number: FM-IAD-005	Rev No.: 0	Effective Date: 01/10/2013	Page No.: 1of 1

FACULTY OF INFORMATION AND COMMUNICATION TECHNOLOGY

Full Name(s) of Candidate(s)	Chan Xiao Jing
ID Number(s)	18ACB04918
Programme / Course	Bachelor of Computer Science (Honours)
Title of Final Year Project	An Optimal Near-Infrared Subcutaneous Vein Extraction Using Deep Learning

Similarity	**Supervisor's Comments (Compulsory if parameters of originality exceed the limits approved by UTAR)**
Overall similarity index: 8 % **Similarity by source** Internet Sources: 5 % Publications: 6 % Student Papers: 2 %	
Number of individual sources listed of more than 3% similarity: 0	
Parameters of originality required, and limits approved by UTAR are as Follows: **(i) Overall similarity index is 20% and below, and** **(ii) Matching of individual sources listed must be less than 3% each, and** **(iii) Matching texts in continuous block must not exceed 8 words** ***Note: Parameters (i) – (ii) shall exclude quotes, bibliography and text matches which are less than 8 words.***	

Note: Supervisor/Candidate(s) is/are required to provide softcopy of full set of the originality report to Faculty/Institute

Based on the above results, I hereby declare that I am satisfied with the originality of the Final Year Project Report submitted by my student(s) as named above.

meng

Signature of Supervisor

Name: Dr. Goh Chuan Meng

Date: 20.4.2022

-

Signature of Co-Supervisor

Name: -

Date: -

FYP 2 CHECKLIST

UNIVERSITI TUNKU ABDUL RAHMAN

FACULTY OF INFORMATION & COMMUNICATION TECHNOLOGY (KAMPAR CAMPUS)

CHECKLIST FOR FYP2 THESIS SUBMISSION

Student Id	18ACB04918
Student Name	Chan Xiao Jing
Supervisor Name	Dr. Goh Chuan Meng

TICK (√)	DOCUMENT ITEMS Your report must include all the items below. Put a tick on the left column after you have checked your report with respect to the corresponding item.
	Front Plastic Cover (for hardcopy)
√	Title Page
√	Signed Report Status Declaration Form
√	Signed FYP Thesis Submission Form
√	Signed form of the Declaration of Originality
√	Acknowledgement
√	Abstract
√	Table of Contents
√	List of Figures (if applicable)
√	List of Tables (if applicable)
-	List of Symbols (if applicable)
√	List of Abbreviations (if applicable)
√	Chapters / Content
√	Bibliography (or References)
√	All references in bibliography are cited in the thesis, especially in the chapter of literature review
√	Appendices (if applicable)
√	Weekly Log
√	Poster
√	Signed Turnitin Report (Plagiarism Check Result - Form Number: FM-IAD-005)
√	I agree 5 marks will be deducted due to incorrect format, declare wrongly the ticked of these items, and/or any dispute happening for these items in this report.

*Include this form (checklist) in the thesis (Bind together as the last page)

I, the author, have checked and confirmed all the items listed in the table are included in my report.

(Signature of Student)
Date: 20.4.2022