SEMANTIC SEGMENTATION FOR ABNORMAL LESIONS FROM RETINA IMAGES

BY

SORAWICH TAVEESATITSATIEAN

A PROJECT SUBMITTED IN PARTIAL FULFILLMENT OF THE REQUIREMENTS FOR THE DEGREE OF BACHELOR OF ENGINEERING IN BACHELOR'S DEGREE

KING MONGKUT'S INSTITUTE OF TECHNOLOGY LADKRABANG

ACADEMIC YEAR 2022

SCHOOL OF ENGINEERING

KING MONGKUT'S INSTITUTE OF TECHNOLOGY LADKRABANG

PROJECT CERTIFICATE

Project Title	Semantic Segmentation for Abnormal Lesions from Retina Images
Student Name	Sorawich Taveesatitsatiean ID. 62011249
Degree	Bachelor of Engineering in Biomedical Engineering

Project Advisor Signed: _____

(Dr. May Phu Paing)

Committee Signed: _____

(Assoc. Prof. Dr. Matthew Paul Gleeson)

Committee Signed: _____

(Asst. Prof. Dr. Jamie Alexander O'Reilly)

Committee Signed: _____

(Asst. Prof. Dr. Pimkhuan Hannanta-anan)

Head of Department Signed: _____

(Assoc. Prof. Dr. Sarinporn Visitsattapongse)

Project Title	Semantic Segmentation for Abnormal Lesions from Retina Images
Student Name	Sorawich Taveesatitsatiean
Degree	Bachelor of Engineering in Biomedical Engineering
Project Advisor	Dr. May Phu Paing
Academic Years	2022

ABSTRACT

Retinal lesions are a severe disease that must be detected at an early stage to prevent visual loss. Our objective is to develop a convolutional neural network for semantic segmentation in medical images, which can accurately detect lesions at an early stage. We aim to make the predictions easily accessible to physicians by creating a user-friendly web-based application, thereby reducing the time required for diagnosis. The project comprises two main phases. In the first phase, we compare and train different neural network models, including DenseNet, ResNet, and VGGNet, combined with U-Net. We evaluate their performance under controlled hyperparameters such as image size and batch size, using Intersection over Union score as the primary metric. Once we identify the best-performing model, we compare it again with widely used segmentation models such as U-Net and U-Net++. In the second phase, we create a Python script to develop a web-based application that displays the predicted results of the selected model on the input image. We ensure that the application can be accessed on both mobile and host devices.

ACKNOWLEDGEMENTS

I am deeply grateful to my thesis advisor, Dr. May Phu Paing, for providing invaluable guidance and support throughout this project and beyond. I am also thankful to Assoc. Prof. Dr. Chuchart Pintavirooj and Dr. Wibool Piyawattanametha for their valuable feedback and insights.

Also, I would like to thank Yi Zhou, Boyang Wang, Lei Huang, Shanshan Cui, and Ling Shao, who provide the dataset which is the core of this project and keep this project running.

Lastly, I would like to extend my heartfelt appreciation to my family and friends for their unwavering support and encouragement, which has been instrumental in helping me successfully complete this journey.

Sorawich Taveesatitsatiean

TABLE OF CONTENTS

Pages

LIST OF FIGURES

Contents Pages

LIST OF TABLES

LIST OF SYMBOLS/ABBREVIATIONS

Symbols/Abbreviations	Terms
μm	Micrometer
3D	3 Dimension
AI	Artificial Intelligence
AUC-ROC	Area under the Receiver Operating Characteristic Curve
BRB	Blood Retinal Barrier
CNN	Convolutional Neural Network
CNS	Central Nervous System
CT	Computer Tomography
DR	Diabetic Retinopathy
EX	Hard Exudate
FGADR	Fined-Grain Annotated Diabetic Retinopathy
GIF	Graphic Interchange Format
GPU	Graphic Processing Unit
HE	Hemorrhage
HSV	Hue Saturation Value
in	Inch
IOU	Intersection over Union
IRMA	Intraretinal Microvascular Anomalies
JPEG/JPG	Joint Photographic Expert Group
MA	Microaneurysm
mm	Millimeter
MRI	Magnetic Resonance Image
NV	Neovascularization
PNG	Portable Network Graphic
RAM	Random Access Memory

Symbols/Abbreviations	Terms
ReLU	Rectified Linear Unit
ResNets	Residual Networks
RGB	Terms of red, green, blue color value
ROI	Region of Interest
RPE	Retinal Pigment Epithelium
SE	Soft Exudate
TIFF	Tag Image File Format
WHO	World Health Organization

CHAPTER 1
INTRODUCTION

This chapter includes the background and significance of this research, which introduces the theory of semantic segmentation, along with the research objectives, hypothesis, and the scope of this project.

1.1 Background and Significance in Research

The eyes are the primary sense organs for human vision and play a critical role in our daily lives. Without healthy eyes or vision, we wouldn't be able to carry out our daily tasks effectively. Among various eye diseases, diabetic retinopathy (DR) is one of the most common vision problems globally. Diabetes is a chronic symptom that is caused by abnormal levels of sugar in the bloodstream, which can affect the whole body, especially the nervous and circulatory systems [1]. Based on the World Health Organization (WHO) statistical estimates, more than 347 million people worldwide have diabetes, and it will be ranked as the seventh leading cause of death globally by 2030.

DR is an eye abnormality affected by diabetes, which there is no sign of in the initial stages, but it can be the cause of blindness in middle-aged patients [2]. Retinopathy can cause the blood vessels to clog, leak, and grow arbitrarily, and finally, lead to retinal damage. The asymptomatic nature of DR makes the patients unaware of the disease and hinders the treatment. Therefore, early detection of the DR is crucial to prevent vision loss in DR patients. Once a patient suffers DR, the winds or lesions such as microaneurysms, hemorrhage, exudates, neovascularization, and Intraretinal Microvascular Abnormalities (IRMA) appear on the patient's retina.

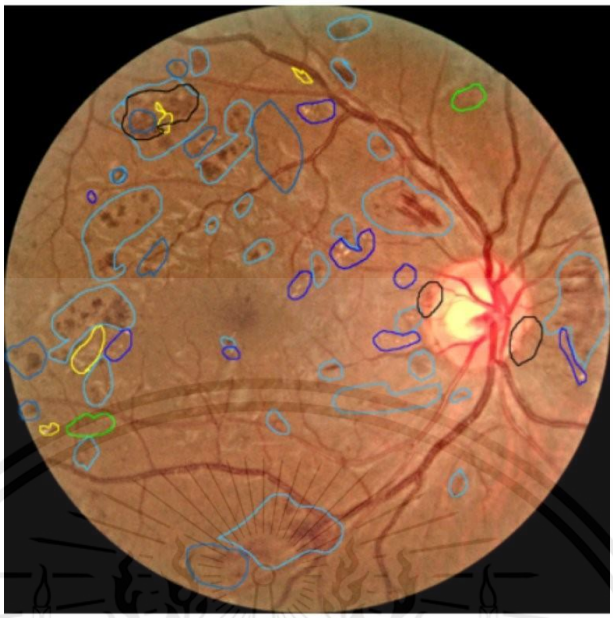

Figure 1: Figure of lesioned fundus with different types of lesions, separated by various colors of contours marked

In clinical practice, a retina fundus exam usually has to be carried out to detect the DR. Then, the ophthalmologist or eye specialists read those fundus images to detect the DR lesions. However, the manual interpretation and diagnosis of DR lesions from the fundus images is an effort and time-consuming task. Moreover, it may sometimes lead to inter or intra-observer variations in diagnosis results.

1.2 Motivation of the Research

One of the main challenges that ophthalmologists encounter during DR diagnosis is to analyze many images and multiple lesions even from one image. This problem hinders a quick and accurate diagnosis. This reason motivates this thesis as an effort to develop an automatic system that can be used as an assistant tool for the eye specialist for the DR lesions detection process. This motivation drives the effort behind this thesis to create an automatic system that can serve as a tool to assist eye specialists in the process of detecting DR lesions.

1.3 Research Gaps

With the advancements in computer vision and deep learning in recent years, much research was reported and developed automatic tools for the diagnosis of DR. However, the majority of such research mainly focused on the classification of DR or the

classification of different eye diseases. Due to the low contrast, irregular shape and size, and blurred edges of the DR lesions, segmenting those lesions from the fundus images still remains as a challenging task. In an effort to solve this problem, our thesis presents an automatic segmentation scheme of DR lesions from fundus images. We will apply semantic segmentation using deep learning models. Unlike image classification which will classify the whole image, semantic segmentation is the process to classify or understand DR lesions inside the input fundus image [3].

1.4 Research Objectives

The main objective of this thesis is to develop an assistant tool for eye specialists for DR diagnosis. Specifically, we can divide our main objective into two sub-objectives.

- To develop a deep learning model for the semantic segmentation of DR lesions from the fundus images
- To create a web application that can upload fundus images and segment DR lesions.

1.4 Research Hypothesis

To get the objects mentioned in Section 1.4, this thesis will propose different deep learning models for semantic segmentation of DR lesions from the retina fundus images. We will develop a variety of deep learning-based segmentation models such as UNet, DenseUNet, DenseLinkNet, UNet++ and compare the performance. Once we get the model with the best performance, we will develop a web application and deploy the best model that we deployed.

1.5 Research Scope

The scope of our research is limited only for the retina fundus images and only for the binary segmentation, that is the segmentation between DR lesions (include all lesions such as microaneurysms, hemorrhage, exudates, neovascularization, IRMA) and non-DR lesions (such as background and retina). The multi class segmentation of DR lesions will be conducted as a future work.

CHAPTER 2
REVIEW OF THEORY RELATED TO IMAGE SEGMENTATION AND CONVOLUTIONAL NEURAL NETWORK

This chapter discussed the state-of-art of image segmentation and convolutional neural network, along with network architecture during the analysis and design phase of this project. The investigation served two purposes; firstly, to identify digital image processing methods, along with details of digital images used in the project; and secondly, to briefly inform the deep learning method from the root of artificial intelligence to types of convolutional network architecture used in the experimental phase. Finally, to summarize this chapter.

2.1 What is the eye?

The eye acts like a live optical instrument with outer layers that keep the eye light-tight except for the optic axis, which includes the sclera and the pigmented choroid. The lenses are arranged along the optic axis and consist of the cornea, the pupil, the crystalline lens, and the light-sensitive portion. The retina is connected to the brain via the optic nerve, while the other parts of the eye maintain their shapes, and provide nutrition, care, and protection.

Humans have two eyes, located in bony spaces called orbits in the skull. The six extraocular muscles regulate eye movements, and the visible front portion of the eye consists of the sclera, iris, and pupil, covered by a thin layer known as the conjunctiva. The front portion of the eye is also referred to as the anterior section.

Adults have eyes that are very similar in size, with the eyeball measuring around 23.7 mm in sagittal height, 24.2 mm in transverse width, and 22.0-24.8 mm in axial anteroposterior depth, and no difference between sexes or age groups. There is a positive correlation between the transverse diameter and orbital width. The average adult eye measures 24 mm from anterior to posterior and has a capacity of 6 cubic centimeters.

The size of the eyeball increases rapidly from a diameter of 16-17 mm at birth to 22-23 mm by the age of three, reaching its maximum size by the age of 12.

2.1.1 Structure of the eye

The eye is not a perfect sphere and is divided into two regions, anterior and posterior. The cornea, iris, and lens comprise the anterior segment, while the vitreous, retina, choroid, and sclera comprise the larger posterior segment. The cornea is 11.5

mm in diameter and 0.5 mm thick in the center. The remaining five-sixths is made up of the posterior chamber, which has an average diameter of 24 mm. The iris surrounds the pupil and controls how much light enters the eye. For near focus, the ciliary muscle changes the curvature of the lens. Photoreceptor cones and rods in the retina convert light into electrical signals that are conveyed to the brain via the optic nerve.

The eye is made up of three layers that cover different anatomical parts. The cornea and sclera form the outermost layer, the fibrous tunic, which supports the deeper components. The choroid, ciliary body, pigmented epithelium, and iris comprise the middle layer, commonly known as the vascular tunic or uvea. The retina is the innermost layer, and it receives oxygen from both the retinal veins and the choroid's blood vessels.

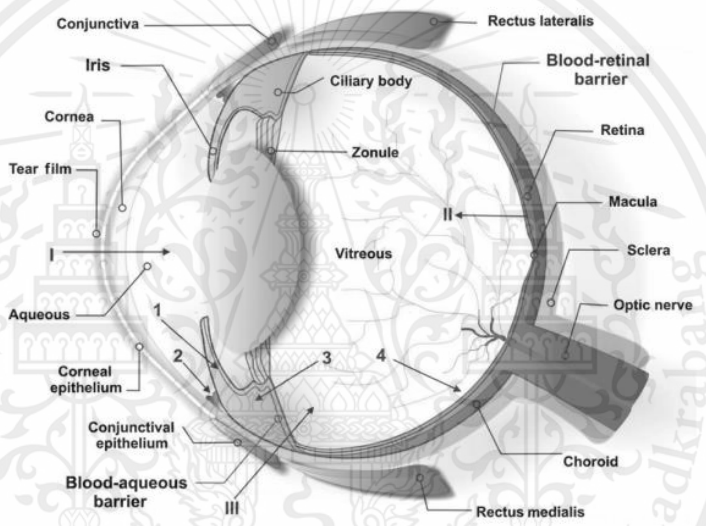

Figure 2: Schematic Illustration of the eyeball

2.1.2 Retina

The retina is a complex structure that plays a critical role in vision. It receives light signals and processes them through a series of chemical and electrical processes, ultimately transmitting nerve impulses to the brain via the optic nerve. The retina is composed of multiple layers of neurons, including photoreceptor cells, which come in two types: rods and cones. Rods are responsible for low-light vision, while cones provide high-acuity vision and color perception. In addition, the photosensitive ganglion cells are responsible for entraining circadian cycles and reflexive reactions,

such as the pupillary light reflex.

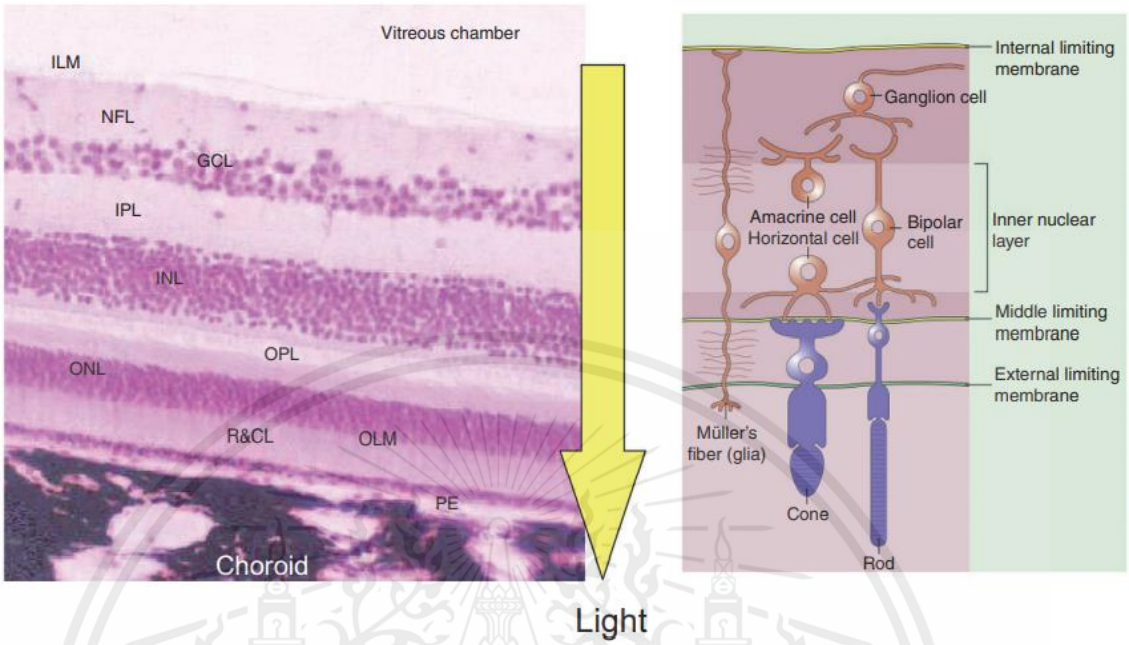

Figure 3: Comparison of cross section and drawing of cells in the retina.

Interestingly, the retina is an extension of the developing brain [4], specifically the embryonic diencephalon. As a result, it is part of the central nervous system and brain tissue, and it is the only part of the CNS that can be examined without invasive methods. The retinal pigment epithelium (RPE) is located between the photoreceptors and the choriocapillaris and performs a variety of critical roles. These include the maintenance of photoreceptor function, such as phagocytosis of photoreceptor debris, regeneration, and pigment production. The RPE is also responsible for retinal adhesion, vitamin A storage and metabolism, and the synthesis of growth factors needed by adjacent tissues. Additionally, it is critical for the proper functioning of the blood-retinal barrier (BRB).

Overall, the retina is a fascinating and complex structure that plays a crucial role in vision. Its interconnectivity with the brain, as well as its unique relationship with the RPE and the BRB, make it an important area of study in the field of ophthalmology and neuroscience.

2.1.3 Fundus

Fundus is the internal surface area of the eye, including the retina, optic disc, macula, fovea, and posterior pole. It can be obtained with an ophthalmoscope.

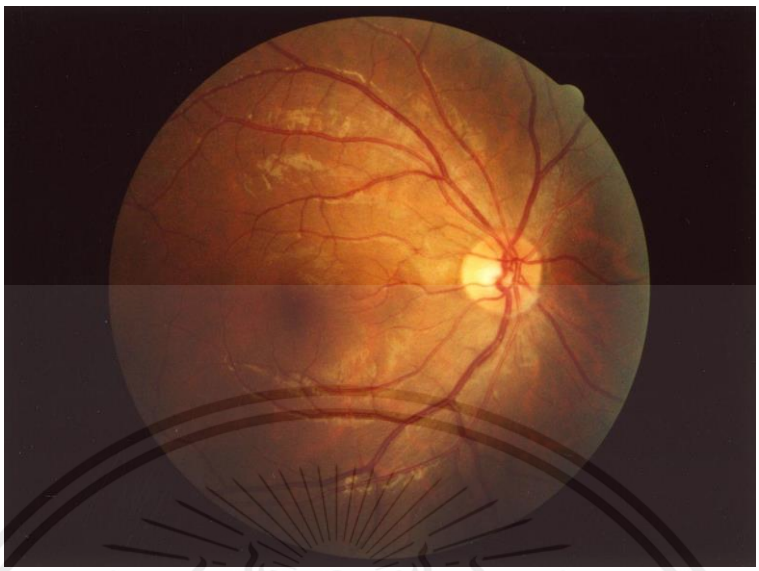

Figure 4: Fundus (eye)

Hemorrhages, exudates, cotton wool patches, blood vessel abnormalities (tortuosity, pulsation, and new vessels), and pigmentation, can all be seen in the eye fundus under a microscope (usually by funduscopy). Hypertensive retinopathy manifests as vascular tortuosities and artery constriction, sometimes known as "silver wiring".

The only area of the human body where microcirculation can be seen in real-time is the fundus of the eye. The optic disc's blood vessels have a diameter of around 150 m, and an ophthalmoscope may view blood vessels as small as 10 μm in size.

2.1.4 What is an ophthalmoscope?

An ophthalmoscope is a medical tool used for internal eye examinations. In 1846, Dr. William Cumming proposed that with the right alignment of a light source and the observer's line of vision, every eye could be made luminous. This concept was presented at the Royal London Ophthalmic Hospital and later became the basis for the modern ophthalmoscope.

With the development of the ophthalmoscope, doctors gained the ability to examine the interior of the eye and diagnose various eye conditions. Today, it is a standard tool used in routine eye exams and is crucial for detecting and monitoring eye diseases such as glaucoma, macular degeneration, and diabetic retinopathy. [5].

Although some attribute the creation of the ophthalmoscope to Charles Babbage in 1847, its value was not appreciated until Hermann von Helmholtz independently reinvented it in 1851.

Figure 5: Early model of Helmholtz Ophthalmoscope, 1851

Before his discovery, ophthalmologists were unable to see the posterior part of the eye and found it difficult to diagnose some groups of eye diseases that resulted in vision loss or dimness. An entirely new era in ophthalmology began in 1851 when the field was suddenly taken by surprise. Men like Albert von Graefe in Berlin, Germany, Edward Jaeger in Vienna, Austria, and William Bowman in London, England, began utilizing the ophthalmoscope as soon as it was invented. Every glance into the eyes turned into a revelation.

2.2 Diabetic Retinopathy

Retina lesions are wounds that grow inside the eyes as a result of illnesses or unusual symptoms; this study will concentrate on lesions brought on by diabetes.

Diabetes is a long-term illness brought on by high blood sugar levels. It can have an impact on many body systems, but particularly the brain and circulatory systems. Diabetes can develop diabetic retinopathy (DR), a disorder of the eyes that has no early warning signs but can lead to blindness in people in their middle years. If untreated, diabetic retinopathy, which is caused by microvascular retinal abnormalities brought on by diabetes, can result in total blindness.

According to the University of Colorado, Anschutz Medical Campus, there are 8 different types of lesions [6]. But in this project, there are 6 different types of diabetic retinopathy lesions used as datasets, which are:

1. Microaneurysms (MA): the earliest visible manifestation of diabetic retinopathy, mostly found as tiny red dots on fundus images

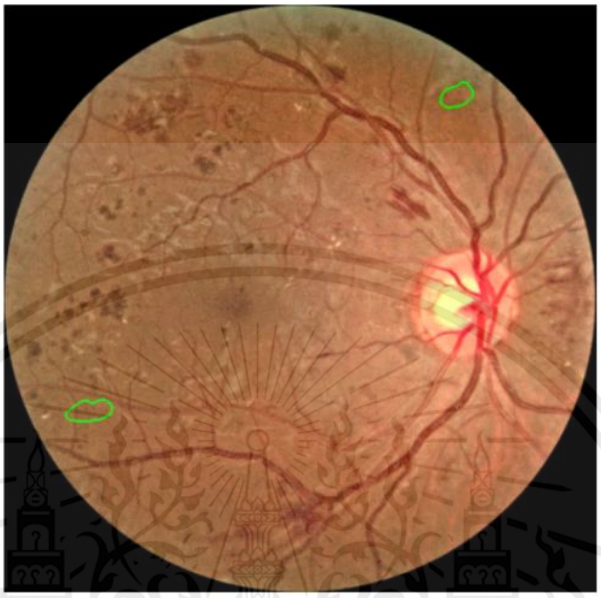

Figure 6: Figure of retina fundus with an area of microaneurysms marked as a green contour.

2. Hemorrhages (HE): abnormal bleeding within the delicate blood vessels of the retina, which appeared as dark marks on the image

Figure 7: Figure of retina fundus with an area of hemorrhages marked as a light blue contour.

3. Soft Exudates (EX): Cotton wool spots on the image, yellowish white color

Figure 8: Figure of retina fundus with an area of soft exudates marked as a yellow contour.

4. Hard Exudates (SE): more yellow in color than soft exudates, caused by lipid residue from damaged capillaries

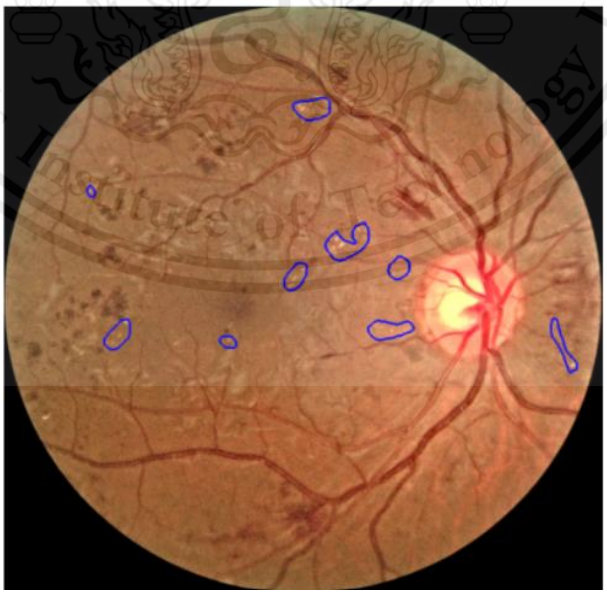

Figure 9: Figure of retina fundus with an area of hard exudates marked as a blue contour.

5. Intraretinal Microvascular Anomalies (IRMA): the traces of capillaries' abnormality

Figure 10: Figure of retina fundus with an area of intraretinal microvascular anomalies marked as a dark cyan contour.

6. Neovascularization (NV): trail of new forming blood vessels, main causes of visual loss in diabetic retinopathy

Figure 11: Figure of retina fundus with an area of neovascularization marked as a black contour.

2.3 Digital Image

A digital picture, which is defined from a function as a two-dimensional image [7], represents data that has both spatial and intensity formation [8], Instead of waiting for films to wash out, it can form in coordinate systems termed "pixel" from the signal received in input and transmitted out to the computer or camera in digital format. Similar to film photos, digital images need to be processed in order to produce better output for usage in other applications. This is where digital image processing comes into play.

Digital images are used in diagnosis for a variety of conditions and purposes in the medical industry, most notably X-ray imaging, medical computer tomography (CT), and magnetic resonance imaging (MRI) [7].

2.3.1 Types of Digital Images

In everyday life, we commonly distinguish between black-and-white and color images. However, in the realm of image processing, there exists more than just these two types. The choice of image processing technique depends on the specific scope and purpose of the task at hand.

Black and white image

Black and white images use a level of brightness of gray color level in each pixel, called “grayscale” to create the image. Normally, the scale of gray color is labeled from 0 – 255, which according to one pixel can represent 8-bit of memory.

- Color image

Similar to black and white photographs, but instead of using a single gray level, a color image uses three different color levels that are stacked together, with red, green, and blue (RGB) typically serving as the primary colors [6]. The amount of memory required to create each pixel will be three times greater than that of a black-and-white image when three sets of color levels are stored as 24-bit per pixel.

- Binary image

The binary picture displays the image by using only "on-off" or "0 - 1," the image will only have black or white color instead of varying shades of gray or other colors.

- Indexed color image

Each pixel's value corresponds to a color from a palette or an index that fixes the output color. This form of image can only be stored in memory which is equivalent to black and white.

2.3.2 Colors of Digital Image

Along with types of a digital image, there will be various types of colors composed into images depending on input which send signals in the format of colors.

- RGB

On a digital color image, the default format uses pixel-level color coding. The color of each pixel is often determined by Color Code, which uses six base-16 digits, one every two numbers, to represent values from 0 to 255. White color (255,255,255) has the color code #FFFFFF, while black color (0,0,0) has the color code #000000.

- HSV

The hue and saturation levels in hue saturation value (HSV) are divided into two categories. Hue is used to denote "color" at any shade level; the RGB value ratio remains constant. Pure color is denoted by saturation [9].

- Intensity/Brightness

Specify the pixel's brightness. Mostly using Hue and Saturation to translate RGB color codes into colors in another system.

2.3.3 Formats of Digital Image

An image file format is a standardized method of encoding and storing digital images. Common formats include JPEG, PNG, GIF, and BMP, each with its unique features and limitations. Historically, image formats were primarily designed to store 2D photographs, but advancements in technology have enabled the development of formats capable of handling 3D images and animations.

Image file formats may use lossy or lossless compression algorithms to reduce file size while maintaining image quality. Lossy compression results in some loss of image detail but produces smaller file sizes, making it ideal for web-based images where speed is critical. Lossless compression algorithms, on the other hand, retain all image details but produce larger file sizes.

Vector image formats, such as SVG and AI, are commonly used in graphic design software as they allow for infinite scalability without loss of resolution. Some image file types, such as PNG and GIF, support transparency, allowing for more flexible design options. Understanding the strengths and weaknesses of each image file format is essential to ensure the best results for your specific application.

List of digital image formats.

- Portable Network Graphic (PNG)

Portable Network Graphics is referred to as PNG. Since the GIF patent was held by a single business and no one else was interested in paying license fees, it was developed as an open format to take the place of GIF. Additionally, it enables a wider color gamut and improved compression.

- Joint Photographic Experts Group (JPEG)

JPEG, created by the Joint Photographic Experts Group, is a highly popular file format for compressed images, capable of efficiently storing large amounts of data in small file sizes. This format is widely used in digital cameras due to its ability to maximize photo storage on a single card, outperforming other formats.

To make a JPEG file smaller, a technique known as "lossy" compression is used, which results in some visual detail being lost during the compression process.

Since they provide a compact file that is simple to load on a web page and also has good visuals, JPEG files are typically used for images on the internet.

Line drawings, logos, and images don't look good in JPEG files because of the effect that the compression causes on jagged lines instead of straight ones.

- Graphic Interchange Format (GIF)

GIF, short for Graphic Interchange Format, is a file format that is ideal for web use due to its narrow color gamut. It allows for a lossless reduction of photo size and can be used to create animations. However, it is not suitable for printing due to the limited color range.

- Tag Image File Format (TIFF)

TIFF stands for Tag Image File Format and produces large file sizes due to its uncompressed nature. TIFF images contain a lot of detailed image data and are versatile in terms of color, size, and content. It can be used in various fields such as print design and photo editing, making it a popular file type for page layout tools like Quark and InDesign as well as photo applications like Photoshop.

2.4 Digital Image Processing

Image processing is a critical field that involves transforming digital images into useful outcomes for various applications. While there are various types of image processing, this project will focus on image segmentation, which involves dividing an image into different segments to identify specific features or objects. This process is essential for numerous applications such as object recognition, medical imaging, and computer vision.

Digital image processing has revolutionized the field of image processing by providing several advantages over analog image processing. The multidimensional nature of digital images enables a broader range of algorithms to be applied, preventing issues such as noise accumulation and distortion during processing. The field's growth and development have been shaped by three main factors: the advancement of discrete mathematics theory, the development of computers, and the growing demand for image processing applications across various industries like environment, agriculture, military, industry, and medical science.

2.4.1 Image Segmentation

Image segmentation is an important stage in image processing that includes splitting an image into segments for data analysis and categorization. Image segmentation, as opposed to image classification, which labels the entire image, and classifies the detected items inside the image. A feature or computed characteristic, such as color, intensity, or texture, is used to compare each pixel in a region. The outlines produced by image segmentation can be used with a stack of photos to make 3D reconstructions using interpolation methods such as marching cubes, which are commonly used in medical imaging.

One of the most fundamental image segmentation techniques is thresholding, which transforms a grayscale image into a binary image depending on a clip-level or threshold value. The maximum entropy approach, balanced histogram thresholding, Otsu's method, and k-means clustering are all strategies utilized in the industry to select the threshold value. Recent advances in computed tomography (CT) image thresholding entail deriving the thresholds using radiographs rather than the reconstructed image. To evaluate if each pixel belongs to a segment, new methodologies advocate non-linear thresholds based on multi-dimensional fuzzy rules created from fuzzy logic and evolutionary algorithms based on the image lighting environment and application.

2.5 Principle of Artificial Intelligence

Artificial intelligence (AI) has revolutionized the way we tackle mathematically based problems, with its ability to harness the power of machines to solve them faster than humans. The concept of AI has gone through several phases of optimism, disappointment, and funding loss, followed by renewed efforts, successes, and more investment. AI researchers have tried and abandoned various approaches, including modeling human problem-solving, formal logic, large knowledge libraries, and animal behavior imitation. In the first two decades of the twenty-first century, machine learning has dominated the field, demonstrating its effectiveness in solving complex problems in both industry and academia.

The importance of AI resides in its ability to surpass humans in some activities and deliver previously unforeseen insights into firms' operations. AI technology can do jobs swiftly and with few errors, particularly in repeated, detail-oriented tasks. Machine learning and deep learning are two methods of artificial intelligence that use massive amounts of data to make predictions. The prospect of creating intelligent artificial entities, on the other hand, has prompted philosophical debates concerning the mind and the moral implications of such creations. If AI is not directed toward useful goals, it may one day represent an existential threat to humans, sparking ongoing discussions among computer scientists and philosophers.

2.5.1 Deep Learning

Deep learning, which is the subset of artificial intelligence, differs from machine learning in that it relies on several representations and transformations to achieve the desired outcome, and can be applied to a wide range of complex issues [10].

Deep learning is a type of artificial intelligence that is characterized by its use of multiple layers in a neural network, hence the term "deep." Earlier research found that a neural network with one hidden layer of unbounded width and a non-polynomial activation function can act as a universal classifier, whereas a linear perceptron cannot. However, modern deep learning focuses on an unbounded number of layers with limited sizes, making it more practical and efficient to implement while still retaining theoretical universality under favorable circumstances. To achieve effectiveness, trainability, and interpretability, deep learning layers can be varied and differ significantly from biologically inspired connectionist models.

Deep learning is a revolutionary approach to machine learning that involves teaching machines how to transform input data into increasingly abstract and composite representations through a series of hierarchical layers. For instance, in image recognition, the first input layer could be a matrix of pixels, and the first representational layer could abstract the pixels and encode edges. The subsequent layers could then build upon this abstraction and encode increasingly complex features such as noses, eyes, and faces. Importantly, the deep learning process can autonomously choose which features to organize at each level, and varying layer counts and widths can lead to different levels of abstraction. Today's deep learning models are primarily

based on artificial neural networks, with convolutional neural networks (CNNs) being a widely used architecture, built using a range of mathematical equations to generate the intricate patterns necessary for sophisticated data analysis.

2.6 Convolutional Neural Network

Artificial neural networks are composed of three types of layers, convolutional, pooling, and fully connected layers [11] and each type has a different purpose and structure, but the most important will be convolutional layers.

Convolutional neural networks (CNNs) are a type of multilayer perceptron that has been modified. Every neuron in one layer is connected to every neuron in the next layer in fully connected networks, which can lead to overfitting. To address this issue, CNNs assemble patterns of increasing complexity using a hierarchical structure in the data. They accomplish this by imprinting their filters with smaller and simpler patterns. This unique regularization method involves reducing connection and complexity.

In CNNs, the convolutional layer plays a critical role in how the model operates. The kernels and activations are the key components of this layer. The kernels perform the calculations, while activations specify the locations or positions where the calculations are made. The pooling layer also helps to reduce the input data's dimension by using max-pooling on a 2*2 scale. Finally, the fully connected layer arranges the neurons for calculation in the network.

In Addition, a "block" typically refers to a group of neural network layers that are combined to form a building block of a larger network architecture. The layers within a block are usually structured in a particular way to perform a specific function, such as feature extraction or downsampling.

For example, a common type of block in convolutional neural networks is the "convolutional block", which typically consists of a convolutional layer followed by a batch normalization layer and a rectified linear unit (ReLU) activation function. This combination of layers can be stacked together multiple times to create a deeper network architecture.

Another example is the "residual block", which is commonly used in residual networks (ResNets) to help mitigate the vanishing gradient problem that can occur in very deep networks. A residual block typically consists of two convolutional layers, each followed by batch normalization and ReLU, along with a "shortcut" connection that bypasses one or more layers in the block. This shortcut connection allows gradient information to be propagated more easily through the network, which can help improve the overall performance.

2.7 Network Architecture

Deep Learning's neural networks are a powerful tool for solving complex data-driven problems, inspired by the functioning of the human brain. These networks process input data through multiple layers of artificial neurons to produce the desired output. The applications of neural networks are wide-ranging and include fields such as marketing, healthcare, speech recognition, and face recognition. Through their ability to extract and learn complex patterns, neural networks are a key component of many state-of-the-art machine learning systems.

2.7.1 U-Net

The most popular architecture to use with segmentation techniques can create regions of interest to predict the result. This project uses this architecture as a main core to create deep learning of artificial intelligence.

This architecture is composed of a contracting path and expanding path [12], contracting using padding layers to reduce the size of input while the expanding path uses the reverse method, by unpadding the layer up to the normal scale of the image.

The U-Net design is based on the idea of adding more layers to a standard convolutional network, with pooling operations substituted by upsampling operators. These extra layers boost the output resolution, and the subsequent convolutional layer learns to produce precise output based on the high-resolution input. Many feature channels are present in the U-Net upsampling section, allowing the network to send context information to higher-resolution layers. This results in a fairly symmetric U-shaped design to the contracting component. The network does not use fully linked layers, instead relying on the valid section of each convolution and extrapolating the missing context by reflecting the input image. A tiling technique is required to bypass resolution constraints imposed by the GPU and RAM when applying the network to huge images.

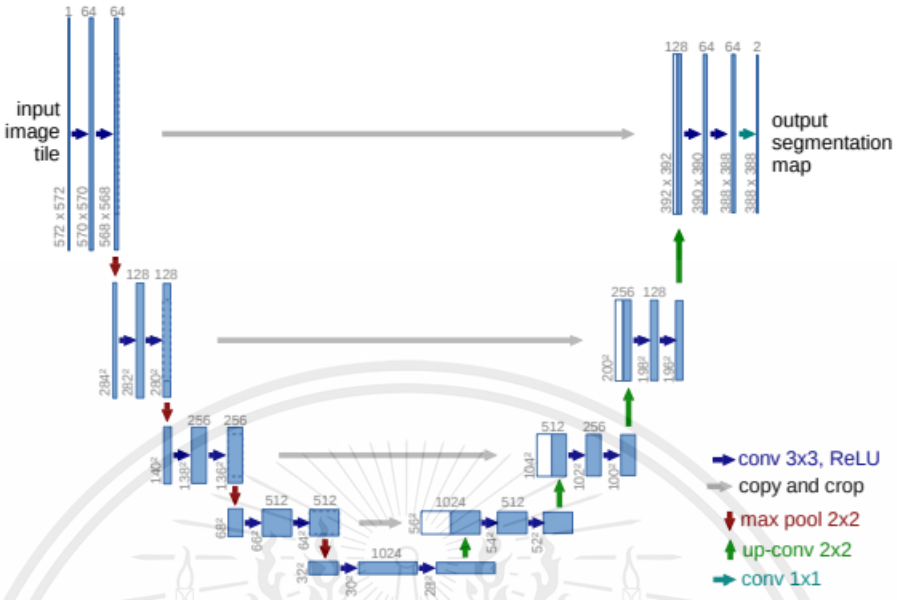

Figure 12: U-net Architecture [12]

The U-shaped topology of the network consists of two parts: the contracting path and the expansive path. The contracting path is similar to a regular convolutional network, performing repeated convolutions, rectified linear unit (ReLU), and max pooling operations to reduce geographical information and increase feature information. In contrast, the expansive path combines feature and spatial data using a series of up-convolutions and concatenations with high-resolution features from the contracting path. The u-shaped structure of the network allows it to retain spatial accuracy while capturing high-level data.

2.7.2 ResNet

Residual neural networks (ResNets) are an essential breakthrough in deep learning architecture. They were developed as a variation of HighwayNet, which was the first functionally complete, deep feedforward neural network with hundreds of layers. ResNets have the capability to bypass some of the layers by using skip connections or shortcuts. This mechanism allows them to be much deeper than earlier neural networks without the risk of vanishing gradients. In typical ResNet models, double- or triple-layer skips containing ReLU nonlinearities are used with batch normalization in between. DenseNets are another type of neural network that includes numerous parallel skips. Non-residual networks are also known as plain networks, which are often compared to ResNets to highlight their efficiency.

Due to its shortcut in convolutional layers with the same size of kernels as can be shown in Figure 2 below, Residual Network, or ResNet, is the opposite of a typical neural network [13].

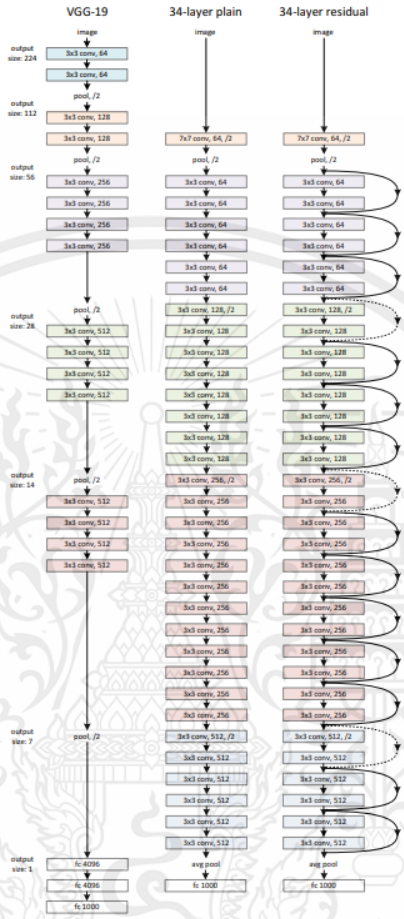

Figure 13: Comparison of VGG-19 (left), ImageNet (center), ResNet (right)

This model focuses more on image identification than segmentation, however it can recognize and segment image data by combining U-Net with ResUNet. ResNet34 and ResNet50 will be the ResNet structures employed in this research.

2.7.3 VGG16

The VGG16 model, described in "Very Deep Convolutional Networks for Large-Scale Image Recognition" by K. Simonyan and A. Zisserman of the University of Oxford, is a convolutional neural network that has shown substantial performance in image recognition. In the top five tests on the ImageNet dataset, which contains over 14 million images classified into 1000 classes, the model obtained an accuracy of

92.7%. VGG16 was submitted to ILSVRC-2014 and quickly became a popular model in the field. The model improves AlexNet by replacing huge kernel-sized filters in the first and second convolutional layers with numerous 3x3 kernel-sized filters successively. VGG16 was trained for weeks using NVIDIA Titan Black GPUs [14].

2.7.4 DenseNet

A DenseNet is a special kind of convolutional neural network that employs dense connections between layers. In Dense Blocks, all layers with similar feature-map sizes are directly connected to one another, resulting in each layer receiving extra inputs from all preceding layers and transmitting its own feature-maps to all subsequent layers. This increases the feed-forward process's efficiency and precision. Compared to standard convolutional neural networks, where each layer is only connected to its following layer, DenseNets have a greater degree of connectivity, which can result in better performance in a variety of computer vision applications.

The "bottleneck" component of this architecture screens the image before it is passed on to the remainder of the model [15]. DenseNet focuses on recognition, just like VGG and ResNet, therefore it may be used with U-Net to accomplish both functions of DenseNet and U-Net, calling Dense-U-Net.

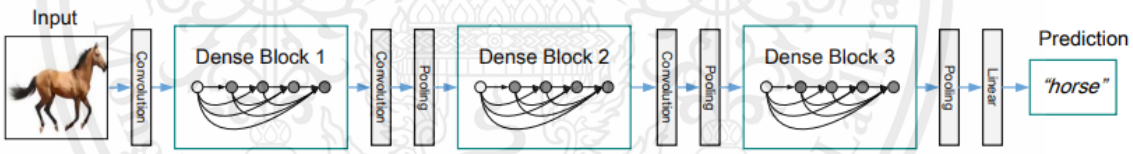

Figure 14: Rough Architecture of DenseNet [13]

2.7.5 U-Net++

The U-Net++ architecture is an extension of the U-Net architecture, which is a popular convolutional neural network (CNN) architecture used in image segmentation tasks. U-Net++ aims to improve segmentation accuracy even further by supplementing the original U-Net architecture with additional nested and skip pathways. These pathways are intended to capture more fine-grained information at various scales, which can improve the network's ability to segment objects of varying sizes and shapes [16].

U-Net++ extends the original U-Net architecture by incorporating nested and dense skip pathways. The nested skip pathways are made up of a series of nested U-Net blocks, each of which is a miniature U-Net architecture with its own contracting and expanding paths. Each block's output is fed as input to the next block, and the final

output is concatenated with the previous block's output. This enables the network to capture multi-scale contextual information and improve segmentation accuracy.

The dense skip pathways are made up of dense blocks, each of which is made up of a series of densely connected convolutional layers. Similarly to the nested skip pathways, the output of each dense block is concatenated with the output of the previous block. The dense skip pathways enable the network to capture more fine-grained information at various scales, potentially improving segmentation accuracy.

Overall, U-Net++ is an extension of the original U-Net architecture that employs nested and dense skip pathways to capture more fine-grained information at various scales, enhancing the network's ability to accurately segment objects of varying sizes and shapes.

2.7.6 LinkNet

LinkNet is a convolutional neural network (CNN) architecture that was proposed in 2017 for the purpose of semantic segmentation in images. It is designed to achieve high segmentation accuracy with a relatively low computational cost [17].

LinkNet's architecture is similar to U- Net's, with an encoder-decoder structure and skip connections between the corresponding encoder and decoder layers. LinkNet, on the other hand, employs a series of residual connections in its encoder and decoder modules, allowing it to efficiently propagate gradients and learn deep features while avoiding the vanishing gradient problem. Furthermore, LinkNet employs a novel "link" module that connects the encoder and decoder pathways and allows them to more effectively share information.

2.8 Region of Interest and performance matrices

The Region of Interest (ROI) is a fundamental concept in image processing, especially in medical imaging. It is a specific region within an image that requires further analysis and processing to achieve a higher level of accuracy in diagnosis. For instance, in medical imaging, the ROI can be a tumor or the endocardial border in an image to evaluate cardiac function or calculate the size, respectively. Accurate identification and scoping of the ROI are crucial to achieve precise diagnosis.

To evaluate the performance of the model, various performance metrics can be used, including the percentage of ROI selected during model building. The selection of the optimal ROI percentage value can significantly improve the accuracy of the model by focusing the processing on the area of interest. The Dice coefficient, a statistical measure of the overlap between two sets of data, is commonly used as the validation value in model evaluation. Additionally, other metrics such as sensitivity, specificity,

and the area under the receiver operating characteristic curve (AUC-ROC) are used to evaluate the model's performance.

Several methods can be used to scope the ROI, including the Intersection over Union and the Dice similarity methods. These methods calculate the overlap between two sets of data, which can be used to evaluate the model's performance. The Intersection over Union method calculates the ratio of the area of overlap to the area of union, while the Dice similarity measures the similarity between two sets of data. These methods are widely used in evaluating the performance of models in image processing tasks, especially in medical imaging, where accurate diagnosis is critical.

2.8.1 Intersection of Union

Intersection over Union (IOU) is a metric that calculates the overlap between predicted and ground truth bounding boxes to assess the accuracy of object identification algorithms. The IOU score is derived by dividing the intersection area of the two boxes by their union area. An IOU score of 1 indicates a perfect match between the expected and ground truth bounding boxes, whereas a value of 0 indicates no overlap.

In object detection applications, IOU is commonly used as a threshold for non-maximum suppression (NMS), a technique used to remove redundant bounding boxes that overlap with higher-scoring boxes. Bounding boxes with an IOU score below a certain threshold are discarded, while those with a higher score are retained. The selection of the threshold value depends on the specific task and dataset and can affect the performance of the model.

However, it still includes a bounding box and value-maximizing gap [18].

Normally, the equation of IOU will be;

$$IOU = \frac{|A \cap B|}{|A \cup B|}$$

2.8.2 Dice similarity

A statistic called the dice similarity coefficient is used to compare two samples. The botanists Lee Raymond Dice and Thorvald Srensen separately created it, publishing their findings in 1945 and 1948, respectively. The simplest equation will be:

$$D = \frac{2TP}{2TP + FP + FN}$$

Dice similarity can perform on a smaller scale than IOU, but if the mask is larger than the images, IOU can be performed better.

2.9 Chapter Summary

We introduced the background of image segmentation and retinal lesions in Chapter 1.

Digital images, image segmentation, artificial intelligence, and region of interest were the different categories used in this chapter to group the state-of-the-art. Observations were made on the systems examined in the section on network architecture, and a summary of the applicability of the state-of-the-art to semantic segmentation was provided.

The construction of convolutional neural network models is discussed in the following chapter. This system is intended to test various model architectures on the same dataset to compare the results and determine which model architecture is best for this project.

CHAPTER 3
METHODOLOGY

3.1 Introduction

In chapter 2, we identified that retina lesion is lethal for humans if it doesn't take any actions until it is too late.

This chapter will describe the design of deep convolutional neural network, a system that is used for detecting the lesions on retina fundus. First, the chapter describes the project's design methodology, and the system requirements. A proposed solution and discussed, followed by a process of solution.

3.2 Design Methodology

The design methodology consists of two parts: constructing the neural network and creating the application. The construction of a neural network requires time to test the compatibility of the dataset we have and the neural network we constructed with different models and types of measuring the function tools. Also, creating the application will be another process we have to take care in to test run its function before officially launching.

3.2.1 Designing and testing the neural network

To construct a convolutional network, we need a python-based interpreter for creating the program used as the base of this project. This project uses Jupyter notebook as the interpreter, with its specialty to run the program in "cells" to separately run the code in sections which can adjust the code in specific points and continually run the process without restarting the running process.

The convolutional network uses U-Net based due to this network architecture is specialized on image segmentation, but still needs more adjustment to make the network working better, which this project will prepare different construction for comparing the performance.

By testing the network architectures to check their performance, then selecting the best performance one to combine and compare with original U-Net, U-Net++ to create the model which ideally detects the area of lesions precisely.

After getting the desired convolutional network, the testing process will come to tune it up for better performance.

3.2.2 Designing the Application

After the model selecting and comparing process is complete, the next step is to create an application to perform the detection of lesions from the image which are inserted as input and to display the predicted image with color lines drawn on the image as a prediction from the model.

Our challenge in creating the application is: it can upload the image easily, it can use it easily, and it must be compatible with any operating system, which makes the decision to create a web-based application to perform the prediction.

Streamlit is an open-source Python library that can create web-based applications easily and can be compatible with machine learning and data science [19], it cannot use with Jupyter Notebook in the present condition, making Microsoft Visual Studio Code came for creating web-based applications from the python programming language.

3.2.3 Dataset used

This project will use the dataset of a large-scale Fine-Grained Annotated Diabetic Retinopathy (FGADR) created by Yizhou, with segmentation set at 1842 images set composed of image and lesion masks that appeared on that fundus image, containing 1,842 images with both pixel-level lesion annotations and image-level grading labels. The lesions include microaneurysms (MA), hemorrhages (HE), hard exudates (EX), soft exudates (SE), intraretinal microvascular abnormalities (IRMA), and neovascularization (NV) [20].

3.3 Interesting Problems

Retina lesions are hard to detect in the early stage, which is a lethal sign of abnormality in vital organs of fundamental human senses. If we can detect and treat it earlier, it might help people suffer less.

3.4 Proposed Solution

Creating the deep learning type of artificial intelligence which can receive the image data of the retina fundus to detect the area, which is possible to be the lesion, for physicians to diagnose the level of risk before the treatment begins.

Also, creating a web-based application will be easy to use and reduce the time to understand the mechanism, and can diagnose faster and more precisely.

3.5 Summary

This chapter described the high-level requirements and design of a system that the neural network should work. The chapter started by describing the process of construction. The proposed solution was then discussed in the process of solution followed by the creation of the application.

The result of the neural network training is covered in further detail in Chapter 4 which describes the implementation of the application.

CHAPTER 4

EXPERIMENTAL RESULT AND DISCUSSION

4.1 Introduction

Chapter 3 described the design and the implementation of network architecture, which is the core of these experiments and projects. In this chapter, we present a testing method and its result that show the comparison and region of interest predicted from different architectures, The chapter is organized as follows: Section A: introducing system and architectures used for semantic segmentation, along with dataset editing, section B: comparing the architecture models from its results.

The results of the tests are summarized in the result section in Chapter 4.2.

Section A: introducing new architectures used in this project rather than U-Net and U-Net++, which are composite architectures.

- Dense U-Net: by using DenseNet architecture as an encoder part of U-Net architecture to perform the training.
- DenseLinkNet: like Dense U-Net architecture but using LinkNet instead of U-Net.
- Res U-Net: by applying ResNet architecture on skip connection of U-Net architecture to adjust the model for more performance.
- VGG U-Net: by using VGGNet architecture as an encoder, then use the decoder part of U-Net.

Due to our main purpose, which is to detect the lesions on retina input, we decided to merge all different types of lesion mask data of images into only lesion masks as shown in the figures below.

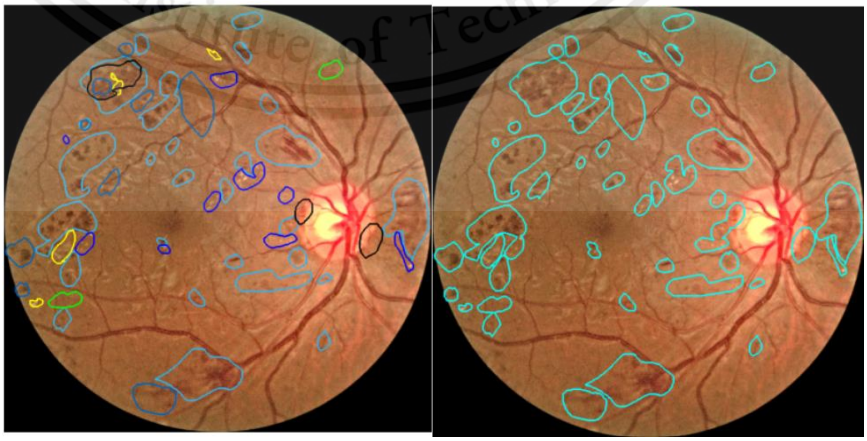

Figure 15: Images showing various lesion masks before (left) and after (right) mask merging process.

Section B: comparing the performance, which divides into 2 sub-parts.

- Part 1: comparing the composite network architectures' performance, which one will be the best among them and more tuning it before entering part 2.
- Part 2: comparing the chosen architecture with the original U-Net architecture, along with U-Net++, and Dense-Link-Net to observe the most compatible architecture for use in the web-based application.

4.2 Result and Discussion

From Section A in Chapter 4.1, which compares the composite architectures for the most compatible in this project, the result will be shown in the table below.

Model	Epochs	Loss	Accuracy	Mean IOU score
Dense U-Net	100	0.0672	0.9795	0.1482
Res U-Net		0.0691	0.9790	0.1392
VGG U-Net		0.0254	0.9923	0.0485

Table 1: Comparison between Dense-U-Net, Res-U-Net and VGG-U-Net architecture models

From the table, it shows that all of them have a high value of accuracy and a very low value of losses, which is a sign of an overfitting phenomenon in training deep learning network architecture. Making accuracy cannot use for validating the model, instead using the IOU score to choose which model can detect the area of lesions predicted as close as the real lesions area, which is Dense-U-Net due to the area this model detected, it will use to compare with original architectures in section B.

In section B, after deciding to use Dense U-Net as the model to compare with other segmentation models, the data used in this comparison will reduce the amount of training sites to lower the overfitting phenomenon in the models. The table below will show the result of each model from training in 40 epochs.

Models	Epochs	Loss	Accuracy	Mean IOU score	Number of images with a score over 0.5	Maximum IOU Score
Dense U-Net	40	0.5520	0.9781	0.3350	222	0.8426
U-Net		0.5342	0.9766	0.3593	265	0.8212
U-Net++		0.6016	0.9736	0.3028	126	0.7250
DenseLinkNet		0.5786	0.9763	0.3169	143	0.7525

Table 2: Comparison between Dense-U-Net, U-Net, U-Net++ and Dense-Link-Net architecture models

From table 2, noticing that the overfitting phenomenon still appeared even the loss of each model become higher from the table 1, but the IOU score is significantly increased in double and in the same range, making it hard to decide the model when observe only with values shown after training, so the figures below will show each model's prediction in the same input.

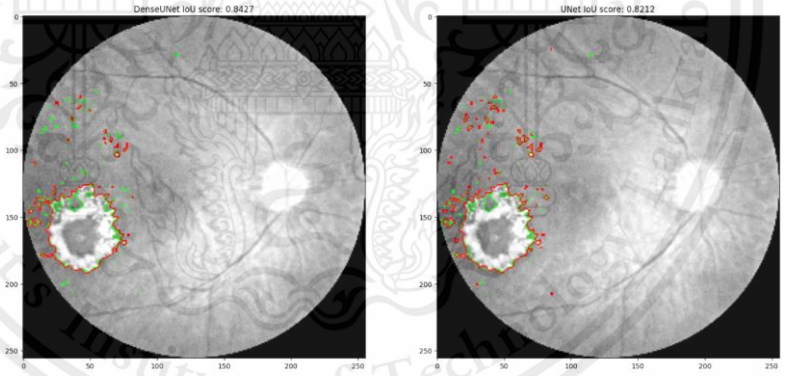

Figure 16: Comparison of prediction of models: Dense-U-Net (left) and U-Net (right)

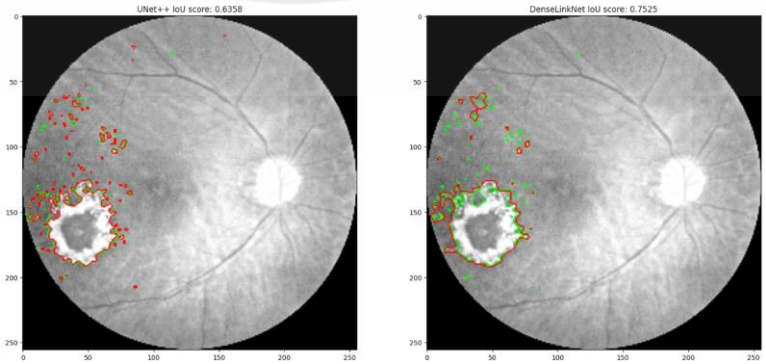

Figure 17: Comparison in prediction of models: U-Net++ (left) and Dense-Link-Net (right)

From the image comparison, finding that Dense U-Net can get a higher score than U-Net but the number of images predicted with a score higher than 0.5 is lower in amount, also with an average score, while both U-Net++ and DenseLinkNet are lower in these three indicators. Making us decide to train both Dense U-Net and U-Net again at 100 epochs to observe the difference.

The result of the training is shown in the table below.

Models	Epochs	Loss	Accuracy	Mean IOU score	Number of images with score over 0.5	Maximum IOU Score
Dense-U-Net	100	0.5661	0.9809	0.3220	219	0.8225
U-Net		0.5164	0.9770	0.3746	252	0.8192

Table 3: Comparison between Dense-U-Net and U-Net architecture models at 100 epochs

Even at 100 epochs, U-Net is still outperforming Dense-U-Net in both number of images and average score, while Dense-U-Net still holds the highest score in prediction of each image. By comparing these indicators, we decided to put both models into the web application and make it can be manually chosen by the user to use which models in prediction.

4.3 Summary

This chapter introduced network architecture models used in this experiment and further procedure in further than Chapter 3 described about the models, section A showing the comparison between composite models and the choice chosen.

Section B shows another comparison in composite models with the widely used network architecture models used in medical image segmentation and concludes the result decided to use in the application.

CHAPTER 5

CONCLUSION

5.1 Introduction

In this Chapter, we first summarize the work described in this report in Chapter 5.2. Then we draw several conclusions about key parts of the work undertaken in Chapter 5.3, and finally, in Chapter 5.4 we discuss future work and how we see other technologies helping support projects such as this one.

5.2 Summary

Chapter 1 introduced the importance of semantic segmentation, along with the objectives, hypothesis, and scope of this project.

Chapter 2 reviewed the state-of-the-art in digital images, and network architectures. Different models were introduced and described. The potential for using network architectures in deep learning was highlighted.

Chapter 3 describes the design of web-based applications. The separate functions of models that support the requirements were then described in more detail, including training, and comparing the composite architecture models.

Chapter 4 described the implementation training the models which compared with others before using it with the application.

Chapter 5 presents the result of training as a display of application, with predicted results on both desktop and smartphones as two figures below.

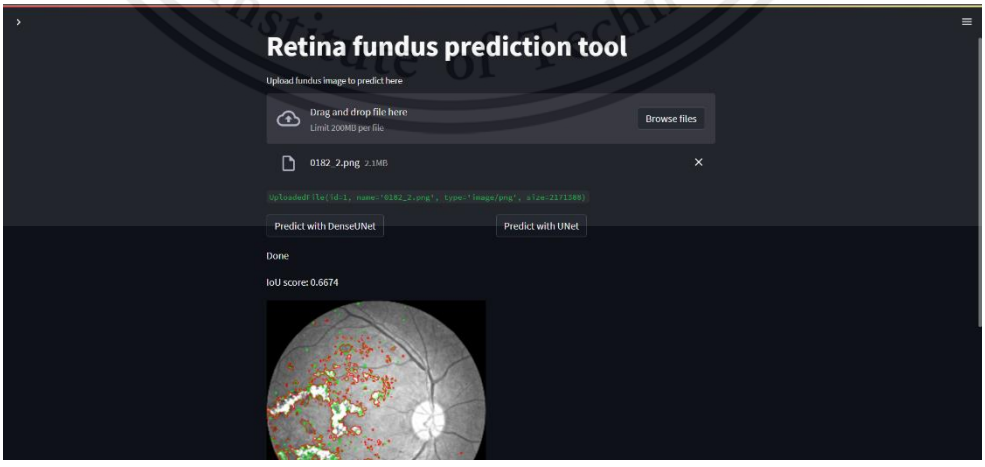

Figure 18: Display of web-based application on computer desktop with predicted images

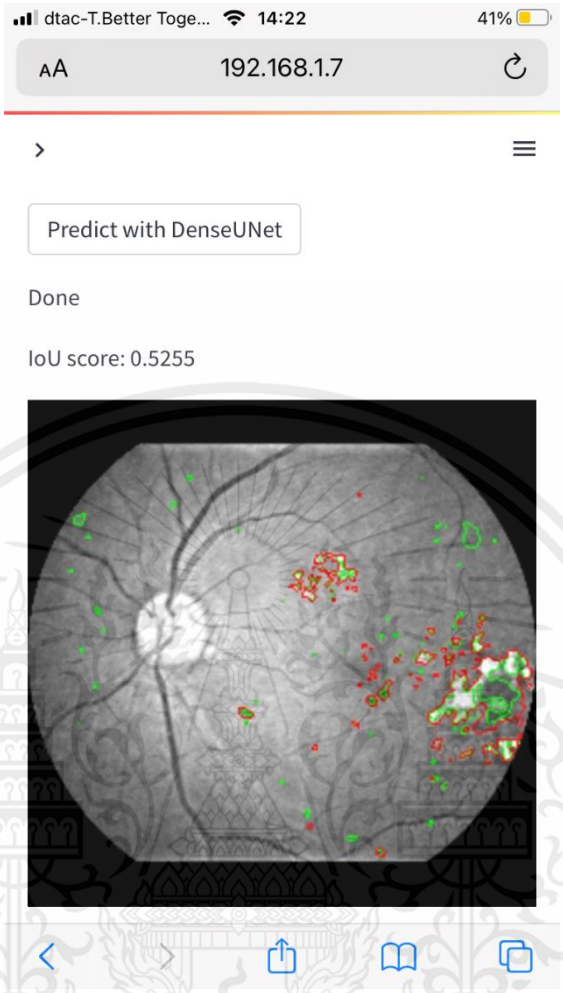

Figure 19: Display of web-based application on smartphone (iPhone) with predicted images

5.3 Conclusions

The primary objective of this project was to develop a network architecture model capable of accurately detecting lesions from the available dataset. Our focus was on achieving a high level of accuracy in lesion detection, as measured by the intersection over union (IOU) score, with a goal of achieving a score of at least 0.6.

To accomplish this goal, we designed and implemented a system with the following capabilities:

• Improved accuracy in IOU score comparisons with the dataset.

• Creating a web-based application to use with smart devices.

These combined capabilities have significant potential for further development and utilization. In Chapter 1, we present the hypothesis that semantic segmentation is

an effective approach for detecting and separating objects in images. To test this hypothesis, we developed convolutional neural network models to predict lesions from input image data.

5.4 Suggestion

There are several potential avenues for further development and utilization of this project. One possibility is to improve the accuracy of lesion detection by further optimizing the IOU score. This could be achieved through various means, such as fine-tuning the network architecture, incorporating additional training data, or exploring alternative evaluation metrics.

Another way to enhance the utility of the project is to connect the web application to a database system, allowing for the collection and storage of new image data for ongoing training and improvement. By continuously updating and expanding the dataset, the model can become more robust and effective over time.

Additionally, it is important to ensure that the web application remains up-to-date with the latest technologies and best practices. Regular maintenance and updates can help to improve performance, fix bugs and errors, and ensure that the application remains compatible with current systems and platforms.

REFERENCES

[1] Ayoub Skouta, Abdelali Elmoufidi, Said Jai-Andaloussi, Ouail Ouchetto, "Hemorrhage semantic segmentation in fundus images for the diagnosis of diabetic retinopathy by using a convolutional neural network," 13 June 2022. [Online]. Available: https://doi.org/10.1186/s40537-022-00632-0. [Accessed 16 October 2022].

[2] M, Usman Akram, Shehzad Khalid, Anam Tariq, Shoab A. Khan, Farooque Azam, "Detection and classification of retinal lesions for grading of diabetic retinopathy," *Computers in Biology and Medicine,* vol. 45, pp. 161-171, 2014.

[3] Dzung L. Pham, Chenyang Xu, Jerry L. Prince, "Current Methods in Medical Image Segmentation," *Annual Review of Biomedical Engineering,* pp. 315-337, 2000.

[4] Colin E Willoughby MD. PhD., Diego Ponzin MD., Stefano Ferrari PhD., Aires Lobo MD., "Anatomy and physiology of the human eye: effects of mucopolysaccharidoses disease on structure and function - a review," *Clinical and Experimental Ophthalmology,* pp. 2-11, 2010.

[5] Keeler, C. Richard, "The Ophthalmoscope in the Lifetime of Hermann von Helmholtz," *Special Article,* pp. 1-8, 2002.

[6] University of Colorado, "Discussion and Pictures of Diabetic Retinopathy Lesions," University of Colorado Anschutz Medical Campus, [Online]. Available: https://medschool.cuanschutz.edu/barbara-davis-center-for-diabetes/patient-care/ophthalmology/discussion-and-pictures-of-diabetic-retinopathy-lesions. [Accessed 19 April 2023].

[7] Ravikumar, Arulmozhi, "Digital Image Processing-A Quick Review," *International Journal of Intelligent Computing and Technology,* vol. 2, no. 2, pp. 16-24, 2019.

[8] Chris Solomon, Toby Breckon, Fundamentals of Digital Image Processing, West Sussex: John Wileyy & Sons, Ltd, 2011.

[9] Jonathan Sachs, "Digital Image Basics," 1996-1999. [Online]. Available: http://www.microscopist.co.uk/wp-content/uploads/2017/04/digital-basics.pdf. [Accessed 20 10 2022].

[10] Yann LeCun, Yoshua Bengio & Geoffrey Hinton, "Deep Learning," in *Nature*, vol. 521, Macmillan Publishers, 2015, pp. 436-444.

[11] Nash, Keiron O'Shea & Ryan, "An Introduction to Convolutional Neural Network," 2 December 2015. [Online]. Available: https://arxiv.org/pdf/1511.08458.pdf. [Accessed 23 October 2022].

[12] Olaf Ronneberger, Phillip Fischer, and Thomas Brox, "U-Net: Convolutional Networks for Biomedical Image Segmentation," 18 May 2015. [Online]. Available: https://arxiv.org/pdf/1505.04597.pdf. [Accessed 25 October 2022].

[13] Kaiming He, Xiangyu Zhang, Shaoqing Ren, Jian Sun, "Deep Residual Learning for Image Recognition," 10 December 2015. [Online]. Available: https://arxiv.org/pdf/1512.03385.pdf. [Accessed 26 October 2022].

[14] Karen Simonyan, Andrew Zisserman, "Very Deep Convolutional Networks for Large-Scale Image Recognition," in *International Conference on Learning Representations*, San Diego, 2015.

[15] Gao Huang, Zhuang Liu, Laurens van der Maaten, "Densely Connected Convolutional Network," 28 January 2018. [Online]. Available: https://arxiv.org/pdf/1608.06993.pdf. [Accessed 28 October 2022].

[16] Zongwei Zhou, Md Mahfuzur Rahman Siddiquee, Nima Tajbakhsh, Jianming Liang, "UNet++: A Nested U-Net Architecture for Medical Image Segmentation," *Deep Learning in Medical Image Analysis and Multimodal Learning for Clinical Decision Support,* pp. 3-11, 2018.

[17] Abhishek Chaurasia, Eugenio Culurciello, "LinkNet: Exploiting Encoder Representations for Efficient Semantic Segmentation," *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition Workshops,* pp. 1799-1808, 2017.

[18] Hamid Rezatofighi, Nathan Tsoi, Jun Young Gwak, Amir Sadgehian, Ian Reid, Silvio Savarese, "Generalized Intersection over Union: A Metric and A Loss for Bounding Box Regression," [Online]. Available: https://openaccess.thecvf.com/content_CVPR_2019/papers/Rezatofighi_Generalized_Intersection_Over_Union_A_Metric_and_a_Loss_for_CVPR_2019_paper.pdf. [Accessed October 2022].

[19] Streamlit, "Streamlit Documentation," Streamlit, [Online]. Available: https://docs.streamlit.io/. [Accessed 7 April 2023].

[20] Yi Zhou, Boyang Wang, Lei Huang, Shanshan Cui, and Ling Shao, "A Benchmark for Studying Diabetic Retinopathy: Segmentation, Grading, and Transferability," *IEEE Transactions on Medical Imaging,* vol. 40, pp. 818-828, 2021.

APPENDICES

Appendix A: Python script of DenseNet121-U-Net.

```
import splitfolders
import tensorflow as tf
import segmentation_models

from tensorflow import keras
from tensorflow.keras import backend as K
from tensorflow.keras.preprocessing.image import ImageDataGenerator
from tensorflow.keras.models import Model
from tensorflow.keras.layers import Conv2D, BatchNormalization, Activation, MaxPool2D, Conv2DTranspose, Concatenate, Input
from tensorflow.keras.callbacks import EarlyStopping, ModelCheckpoint, ReduceLROnPlateau, CSVLogger, TensorBoard

from tensorflow.keras.applications import DenseNet121
from tensorflow.keras.metrics import IOU,Recall , Precision
from tensorflow.keras.models import load_model
import matplotlib.pyplot as plt
import numpy as np
import cv2

train_img = 'data/train2/Original_Images'
train_msk = 'data/train2/Merged_Masks'

val_img = 'data/val/Original_Images'
val_msk = 'data/val/Merged_Masks'
```

```
# Prepare data generators
seed = 0
batch_size = 4
img_size = 256

imgDataGenTrain = ImageDataGenerator( rescale=1/255,
                                      data_format = "channels_last",
                                      rotation_range = 180,
                                      width_shift_range = 0.15,
                                      height_shift_range = 0.15,
                                      fill_mode = 'constant',
                                      cval = 0.0,
                                      horizontal_flip = True,
                                      vertical_flip = True)

maskDataGenTrain = ImageDataGenerator(rescale=1/1,
                                      data_format = "channels_last",
                                      rotation_range = 180,
                                      width_shift_range = 0.15,
                                      height_shift_range = 0.15,
                                      fill_mode = 'constant',
                                      cval = 0.0,
                                      horizontal_flip = True,
                                      vertical_flip = True,
                                      preprocessing_function = lambda x: np.where(x > 127, 1,
0).astype(x.dtype))

trainImgGenerator = imgDataGenTrain.flow_from_directory(train_img,
                                        target_size=(img_size,img_size),
```

```
                                    batch_size=batch_size,
                                    shuffle=True,
                                    #color_mode='grayscale',
                                    class_mode=None,
                                    seed=seed)

trainMskGenerator = maskDataGenTrain.flow_from_directory(train_msk,
                                    target_size=(img_size,img_size),
                                    batch_size=batch_size,
                                    shuffle=True,
                                    #color_mode='grayscale',
                                    class_mode=None,
                                    seed=seed)

imgDataGenTest = ImageDataGenerator(rescale=1/255,
                        data_format ="channels_last")

maskDataGenTest = ImageDataGenerator(rescale=1/1,
                        data_format ="channels_last",
                        preprocessing_function = lambda x: np.where(x>127, 1,
0).astype(x.dtype))

valImgGenerator = imgDataGenTest.flow_from_directory( val_img,
                                    target_size=(img_size,img_size),
                                    batch_size=batch_size,
                                    shuffle=False,
                                    #color_mode='grayscale',
                                    class_mode=None,
                                    seed=seed)
```

```
valMskGenerator = maskDataGenTest.flow_from_directory( val_msk,
                                        target_size=(img_size,img_size),
                                        batch_size=batch_size,
                                        shuffle=False,
                                        #color_mode='grayscale',
                                        class_mode=None,
                                        seed=seed)

# Zip together image (x) and mask (y) data generators
trainGenerator = zip(trainImgGenerator, trainMskGenerator)
valGenerator = zip(valImgGenerator, valMskGenerator)

# Check data
train_images = trainImgGenerator.__getitem__(0)
train_masks = trainMskGenerator.__getitem__(0)

val_images = valImgGenerator.__getitem__(0)
val_masks = valMskGenerator.__getitem__(0)


print(train_images.shape)
print(train_masks.shape)

# Write a function for plotting contours on images
def draw_contours(x, y_true, y_pred=None, pred=False):
    image = cv2.merge([x, x, x])
    image = cv2.normalize(image, None, 0, 255, cv2.NORM_MINMAX).astype('uint8')
```

```
    contours = cv2.findContours(y_true.astype('uint8'), cv2.RETR_EXTERNAL, cv2.CHAIN_APPROX_SIMPLE)[0]
    image = cv2.drawContours(image, contours, -1, [0,255,0], 1)
    if pred:
        contours = cv2.findContours(y_pred.astype('uint8'), cv2.RETR_EXTERNAL, cv2.CHAIN_APPROX_SIMPLE)[0]
        image = cv2.drawContours(image, contours, -1, [255,0,0], 1)
    return image


# Display some of the training images with contours
plt.figure(figsize=(8,4))
for i in range(4):
    plt.subplot(1,4,i+1)
    cont_image = draw_contours(train_images[i,:,:,0], train_masks[i,:,:,0])
    plt.imshow(cont_image)
    plt.axis('off')
plt.tight_layout()
plt.show()


import segmentation_models as sm
#from segmentation_models.models import nestnet
sm.set_framework('tf.keras')
sm.framework()



BACKBONE = 'densenet121'
BATCH_SIZE = 4
LR = 0.0001
EPOCHS = 100
```

```
preprocess_input = sm.get_preprocessing(BACKBONE)


n_classes = 1

activation = 'sigmoid'

#create model

model = sm.Unet(BACKBONE, classes=n_classes, activation=activation)

model.summary()


# define optimizer

import tensorflow

optim = tensorflow.keras.optimizers.Adam(LR)


# Segmentation models losses can be combined together by '+' and scaled by integer or float factor

# set class weights for dice_loss (car: 1.; pedestrian: 2.; background: 0.5;)

dice_loss = sm.losses.DiceLoss(class_weights=np.array([1, 2, 0.5]))

focal_loss = sm.losses.BinaryFocalLoss() if n_classes == 1 else sm.losses.CategoricalFocalLoss()

total_loss = dice_loss + (1 * focal_loss)


# actually total_loss can be imported directly from library, above example just show you how to manipulate with losses

total_loss = sm.losses.binary_focal_dice_loss # or sm.losses.categorical_focal_dice_loss


IOU_score = sm.metrics.IOUScore(threshold=0.5)

f1_score = sm.metrics.FScore(threshold=0.5)


metrics = [IOU_score, f1_score,'accuracy']
```

```
config = model.get_config()

custom_objects={"dice_loss": dice_loss, "focal_loss": focal_loss,\
            "binary_focal_loss_plus_dice_loss": total_loss,\
            "metrics":metrics,"IOU_score":IOU_score,"f1_score":f1_score}
with keras.utils.custom_object_scope(custom_objects):
    new_model = model.from_config(config)

# compile keras model with defined optimizer, loss and metrics
model.compile(optim, total_loss, metrics)

callbacks = [keras.callbacks.ReduceLROnPlateau()]

# train model

#Number Of Batches
train_steps = 1473 //BATCH_SIZE
valid_steps = 369  //BATCH_SIZE


history = model.fit(
    trainGenerator,
    steps_per_epoch=train_steps,
    epochs=EPOCHS,
    callbacks=callbacks,
    validation_data=valGenerator,
    validation_steps=valid_steps,
)
```

```
# Save model
model.save('model/DenseUNet_100_false_r82_removed.h5')

# Evaluate model
model.evaluate(valGenerator, steps=(369//batch_size))

# Plot learning curves
plt.figure(figsize=(15, 5))

plt.subplot(121)
plt.plot(history.epoch, history.history['loss'], label='Train')
plt.plot(history.epoch, history.history['val_loss'], label='Test')
plt.axvline(history.epoch[-10], color='k', ls='--', lw=2)
plt.ylabel('Loss')
plt.xlabel('Epoch')
plt.legend()

plt.subplot(122)
plt.plot(history.epoch, history.history['accuracy'])
plt.plot(history.epoch, history.history['val_accuracy'])
plt.axvline(history.epoch[-10], color='k', ls='--', lw=2)
plt.ylabel('Accuracy')
plt.xlabel('Epoch')

plt.subplot(133)
plt.plot(history.epoch, history.history['IOU_score'])
plt.plot(history.epoch, history.history['val_IOU_score'])
plt.axvline(history.epoch[-10], color='k', ls='--', lw=2)
plt.ylabel('IOU')
```

```
 plt.xlabel('Epoch')

plt.tight_layout()
plt.show()

def IOU(result1,result2):
    # IOU calculation
    intersection = np.logical_and(result1, result2)
    union = np.logical_or(result1, result2)
    IOU_score = np.sum(intersection) / np.sum(union)
    return IOU_score

# Display some output images
for n in range(10):
    x = valImgGenerator.__getitem__(n)
    y_true = valMskGenerator.__getitem__(n)
    y_pred = model.predict(x)
    y_pred = np.round(y_pred)


    plt.figure(figsize=(20,20))
    for i in range(4):
        plt.subplot(2,2, i+1)
        cont_image = draw_contours(x[i,:,:,0], y_true[i,:,:,0], y_pred[i,:,:,0], True)
        IOU_score = IOU(y_pred[i,:,:,0],y_true[i,:,:,0])
        plt.title("IOU score: {:.4f}".format(IOU_score))
        plt.imshow(cont_image)

        plt.axis('off')
```

```
plt.show()
```

Appendix B: Python script of comparing the model with the same image samples.

```
from tensorflow.keras.models import load_model
import numpy as np
import matplotlib.pyplot as plt
import cv2


# define optimizer
import tensorflow
#optim = tensorflow.keras.optimizers.Adam(LR)
import segmentation_models as sm
n_classes = 1


# Segmentation models losses can be combined together by '+' and scaled by integer or float factorS
# set class weights for dice_loss (car: 1.; pedestrian: 2.; background: 0.5;)
dice_loss = sm.losses.DiceLoss(class_weights=np.array([1, 2, 0.5]))
focal_loss = sm.losses.BinaryFocalLoss() if n_classes == 1 else sm.losses.CategoricalFocalLoss()
total_loss = dice_loss + (1 * focal_loss)


# actually total_loss can be imported directly from library, above example just show you how to manipulate with losses
total_loss = sm.losses.binary_focal_dice_loss # or sm.losses.categorical_focal_dice_loss


IOU_score = sm.metrics.IOUScore(threshold=0.5)
f1_score = sm.metrics.FScore(threshold=0.5)


metrics = [IOU_score, f1_score,'accuracy']
```

```
def rescaling(x,y):
    return x*y


def IOU(result1,result2):
    # IOU calculation
    intersection = np.logical_and(result1, result2)
    union = np.logical_or(result1, result2)
    IOU_score = np.sum(intersection) / np.sum(union)
    return IOU_score


# Write a function for plotting contours on images
def draw_contours(x, y_true, y_pred=None, pred=False):
    image = cv2.merge([x, x, x])
    image = cv2.normalize(image, None, 0, 255, cv2.NORM_MINMAX).astype('uint8')
    contours = cv2.findContours(y_true.astype('uint8'), cv2.RETR_EXTERNAL,
cv2.CHAIN_APPROX_SIMPLE)[0]
    image = cv2.drawContours(image, contours, -1, [0,255,0], 1)
    if pred:
        contours = cv2.findContours(y_pred.astype('uint8'), cv2.RETR_EXTERNAL,
cv2.CHAIN_APPROX_SIMPLE)[0]
        image = cv2.drawContours(image, contours, -1, [255,0,0], 1)
    return image


import numpy as np
from tensorflow.keras.preprocessing import image


#img_width, img_height = 150, 150
# fix = '1612_3.png'
# fix = '0308_3.png'
# fix = '0120_3.png'
```

```
fix = ['0484_3.png','1368_3.png','0075_1.png','1612_3.png']


dunet = load_model('model/DU_R_40.h5',\
                   custom_objects={"dice_loss": dice_loss, "focal_loss": focal_loss,\
                                   "binary_focal_loss_plus_dice_loss": total_loss,\
                                   "metrics":metrics,"IOU_score":IOU_score,"f1-
score":f1_score})


unet = load_model('model/U_R_40.h5',\
                   custom_objects={"dice_loss": dice_loss, "focal_loss": focal_loss,\
                                   "binary_focal_loss_plus_dice_loss": total_loss,\
                                   "metrics":metrics,"IOU_score":IOU_score,"f1-
score":f1_score})


unetpp = load_model('model/UPP_R_40.h5',\
                   custom_objects={"dice_loss": dice_loss, "focal_loss": focal_loss,\
                                   "binary_focal_loss_plus_dice_loss": total_loss,\
                                   "metrics":metrics,"IOU_score":IOU_score,"f1-
score":f1_score})


dlinknet = load_model('model/DL_R_40.h5',\
                   custom_objects={"dice_loss": dice_loss, "focal_loss": focal_loss,\
                                   "binary_focal_loss_plus_dice_loss": total_loss,\
                                   "metrics":metrics,"IOU_score":IOU_score,"f1-
score":f1_score})


#Comparison

direct = 'C:/Users/geark/Desktop/Senior Project/Jupyter Script Files/FGADR-
Data/dataset/Original_Images/'

direct2 = 'C:/Users/geark/Desktop/Senior Project/Jupyter Script Files/FGADR-
Data/dataset/Merged_Masks/'
```

```
for i in range(len(fix)):
    img = image.load_img(direct+fix[i],target_size=(256,256))
    img = image.img_to_array(img)
    img = rescaling(img,1/255)
    # img = img.reshape((1,img.shape[0],img.shape[1],img.shape[2]))
    img = np.expand_dims(img, axis = 0)

    mask = image.load_img(direct2+fix[i],target_size=(256,256))
    mask = image.img_to_array(mask)
    mask = np.expand_dims(mask, axis = 0)

    y1 = dunet.predict(img)
    y_pred1 = np.round(y1)

    y2 = unet.predict(img)
    y_pred2 = np.round(y2)

    y3 = unetpp.predict(img)
    y_pred3 = np.round(y3)

    y4 = dlinknet.predict(img)
    y_pred4 = np.round(y4)

    x = img
    y_true = mask

    plt.figure(figsize=(20,20))
    plt.suptitle('Comparison in models: IMG = {}'.format(fix[i]),fontsize=20)
```

```
cont_image = draw_contours(x[0,:,:,0],y_true[0,:,:,0],y_pred1[0,:,:,0],True)
IOU_score = IOU(y_pred1[0,:,:,0],y_true[0,:,:,0])
plt.subplot(221)
plt.title("DenseUNet IOU score: {:.4f}".format(IOU_score))
plt.imshow(cont_image)

cont_image = draw_contours(x[0,:,:,0],y_true[0,:,:,0],y_pred2[0,:,:,0],True)
IOU_score = IOU(y_pred2[0,:,:,0],y_true[0,:,:,0])
plt.subplot(222)
plt.title("UNet IOU score: {:.4f}".format(IOU_score))
plt.imshow(cont_image)

cont_image = draw_contours(x[0,:,:,0],y_true[0,:,:,0],y_pred3[0,:,:,0],True)
IOU_score = IOU(y_pred3[0,:,:,0],y_true[0,:,:,0])
plt.subplot(223)
plt.title("UNet++ IOU score: {:.4f}".format(IOU_score))
plt.imshow(cont_image)

cont_image = draw_contours(x[0,:,:,0],y_true[0,:,:,0],y_pred4[0,:,:,0],True)
IOU_score = IOU(y_pred4[0,:,:,0],y_true[0,:,:,0])
plt.subplot(224)
plt.title("DenseLinkNet IOU score: {:.4f}".format(IOU_score))
plt.imshow(cont_image)
```

Appendix C: Python script of web-based application.

```
import streamlit as st
import time

from tensorflow import keras
from keras import models,utils
import numpy as np
import matplotlib.pyplot as plt
import cv2
import os

import segmentation_models as sm
n_classes = 1

st.set_page_config(page_title="Home",layout="centered",initial_sidebar_state="auto"
)

dsk = "C:\\Users\\geark\\Desktop\\"
dft = "C:\\Users\\geark\\Desktop\\Senior Project\\VSCode .py files\\"

# Segmentation models losses can be combined together by '+' and scaled by integer or
float factorS
# set class weights for dice_loss (car: 1.; pedestrian: 2.; background: 0.5;)
dice_loss = sm.losses.DiceLoss(class_weights=np.array([1, 2, 0.5]))
focal_loss = sm.losses.BinaryFocalLoss() if n_classes == 1 else
sm.losses.CategoricalFocalLoss()
total_loss = dice_loss + (1 * focal_loss)

# actually total_loss can be imported directly from library, above example just show
you how to manipulate with losses
```

```
total_loss = sm.losses.binary_focal_dice_loss # or sm.losses.categorical_focal_dice_loss


IOU_score = sm.metrics.IOUScore(threshold=0.5)

f1_score = sm.metrics.FScore(threshold=0.5)


metrics = [IOU_score, f1_score,'accuracy']


dunet = models.load_model(r"C:\Users\geark\Desktop\Senior Project\Jupyter Script Files\Removed Data script\model\DU_R_100.h5",
                          custom_objects={"dice_loss": dice_loss, "focal_loss": focal_loss,
                                          "binary_focal_loss_plus_dice_loss": total_loss,
                                          "metrics":metrics,"IOU_score":IOU_score,"f1-score":f1_score})


unet = models.load_model(r"C:\Users\geark\Desktop\Senior Project\Jupyter Script Files\Removed Data script\model\U_R_100.h5",
                          custom_objects={"dice_loss": dice_loss, "focal_loss": focal_loss,
                                          "binary_focal_loss_plus_dice_loss": total_loss,
                                          "metrics":metrics,"IOU_score":IOU_score,"f1-score":f1_score})



#img_width, img_height = 150, 150

# img = utils.load_img("0037_1.png")

# img = utils.img_to_array(img)

# img = np.expand_dims(img, axis = 0)

mdir = "C:\\Users\\geark\\Desktop\\Senior Project\\Jupyter Script Files\\FGADR-Data\\dataset\\Merged_Masks\\"

# y = history.predict(img)
```

```
def IOU(result1,result2):
    # IOU calculation
    intersection = np.logical_and(result1, result2)
    union = np.logical_or(result1, result2)
    IOU_score = np.sum(intersection) / np.sum(union)
    return IOU_score

# Write a function for plotting contours on images
def draw_contours(x, y_true, y_pred=None, pred=False):
    image = cv2.merge([x, x, x])
    image = cv2.normalize(image, None, 0, 255, cv2.NORM_MINMAX).astype('uint8')
    contours = cv2.findContours(y_true.astype('uint8'), cv2.RETR_EXTERNAL,
cv2.CHAIN_APPROX_SIMPLE)[0]
    image = cv2.drawContours(image, contours, -1, [0,255,0], 1)
    if pred:
        contours = cv2.findContours(y_pred.astype('uint8'), cv2.RETR_EXTERNAL,
cv2.CHAIN_APPROX_SIMPLE)[0]
        image = cv2.drawContours(image, contours, -1, [255,0,0], 1)
    return image

def rescaling(img,size):
    return img*size

def prediction1(img1,filename):
    img = utils.load_img(img1,target_size=(256,256))
    img = utils.img_to_array(img)
    img = rescaling(img,1/255)
    # img = img.reshape((1,img.shape[0],img.shape[1],img.shape[2]))
    img = np.expand_dims(img, axis = 0)
```

```
    mask = utils.load_img(mdir+filename,target_size=(256,256))
    mask = utils.img_to_array(mask)
    mask = np.expand_dims(mask, axis = 0)

    y = dunet.predict(img)
    y_pred = np.round(y)
    x = img
    y_true = mask

    cont_image = draw_contours(x[0,:,:,0],y_true[0,:,:,0],y_pred[0,:,:,0],True)
    IOU_score = IOU(y_pred[0,:,:,0],y_true[0,:,:,0])
    cont_image = cv2.resize(cont_image,(512,512))
    return cont_image,IOU_score

def prediction2(img1,filename):
    img = utils.load_img(img1,target_size=(256,256))
    img = utils.img_to_array(img)
    img = rescaling(img,1/255)
    # img = img.reshape((1,img.shape[0],img.shape[1],img.shape[2]))
    img = np.expand_dims(img, axis = 0)

    mask = utils.load_img(mdir+filename,target_size=(256,256))
    mask = utils.img_to_array(mask)
    mask = np.expand_dims(mask, axis = 0)

    y = unet.predict(img)
    y_pred = np.round(y)
    x = img
    y_true = mask
```

```
    cont_image = draw_contours(x[0,:,:,0],y_true[0,:,:,0],y_pred[0,:,:,0],True)
    IOU_score = IOU(y_pred[0,:,:,0],y_true[0,:,:,0])
    cont_image = cv2.resize(cont_image,(512,512))
    return cont_image,IOU_score


st.title('Retina fundus prediction tool')
#st.markdown('Upload image here')
img1 = st.file_uploader('Upload fundus image to predict here')
st.write(img1)


col1,col2 = st.columns(2)


with col1:
    if st.button("Predict with DenseUNet"):
        with st.spinner("Please wait.."):
            filename = os.path.basename(img1.name)
            img_pred1,score1 = prediction1(img1,filename)
        st.markdown("Done")
        st.write("IOU score: {:.4f}".format(score1))
        st.image(img_pred1)


with col2:
    if st.button("Predict with UNet"):
        with st.spinner("Please wait.."):
            filename = os.path.basename(img1.name)
            img_pred2,score2 = prediction2(img1,filename)
        st.markdown("Done")
        st.write("IOU score: {:.4f}".format(score2))
```

```
        st.write(filename)
        st.image(img_pred2)
st.markdown("Alpha version: 5.0.0")
```