LUNG CANCER DETECTION ON CT SCAN IMAGES WITH DEEP LEARNING METHODS: SUGENO FUZZY INTEGRAL-BASED CNN ENSEMBLE

By
LIM SOONG XIAN

Dissertation submitted in partial fulfilment of the
requirement for the award of the degree of
Bachelor of Computer Science (Software Engineering) with Honours

FACULTY OF OCEAN ENGINEERING TECHNOLOGY AND INFORMATICS
UNIVERSITI MALAYSIA TERENGGANU
2022

DISSERTATION CONFIRMATION AND APPROVAL

This is acknowledged and confirmed that the dissertation entitled: Lung Cancer Detection on CT Scan Images With Deep Learning Methods: Sugeno Fuzzy Integral-Based CNN Ensemble by Lim Soong Xian Matric No.: S54235 have been checked and all the suggested corrections have been done. The dissertation is submitted to the Faculty of Ocean Engineering Technology and Informatics, Universiti Malaysia Terengganu in partial fulfilment of the requirements for the award of the degree of Bachelor of Computer Science (Software Engineering) with Honours.

Authorized by:

…………………………….. Date: 19 July 2022

Main Supervisor

Name: Associate Processor Ts. Dr. Masita @ Masila Binti Abdul Jalil

Official Stamp: PROF. MADYA Ts. DR. MASITA@ MASILA ABD JALIL
Pensyarah Kanan
Fakulti Teknologi Kejuruteraan Kelautan dan informatik
Universiti Malaysia Terengganu

…………………………….. Date: 19 July 2022

PITA's Coordinator

Bachelor of Computer Science (Software Engineering) with Honours

Name: Ts. Dr. Wan Nural Jawahir Binti Hj Wan Yussof

Official Stamp: Ts. DR. WAN NURAL JAWAHIR BINTI HJ WAN YUSSOF
Senior Lecturer
Faculty of Ocean Engineering Technology and Informatics
Universiti Malaysia Terengganu

DECLARATION

I hereby declare that this dissertation is the result of my own research except as cited in the references.

Signature	: ……………………………………
Name	: Lim Soong Xian
Matric No.	: S54235
Date	: 18 July 2022

ACKNOWLEDGEMENTS

At the end of my dissertation, I would like to thank everyone who made this dissertation possible and an enjoyable experience for me, as well as the knowledge shared by the people around the world to widen my horizon

First of all, I wish to express my sincere gratitude to my supervisor, Assoc. Prof. Ts. Dr. Masita @ Masila Abdul Jalil, Faculty of Ocean Engineering Technology and Informatics, for providing me with invaluable knowledge and experience, as well as patient guidance and advice throughout every stage of my research. Her perseverance and support enabled me to successfully complete this thesis.

I am grateful to my friends for their encouragement and help especially to my friend, Soh Eu Jhern who lent me his computer and emotional support to help me riding the wave of the journey of this final year project and for deep learning training.

Finally, I would like also to express my deepest gratitude for the constant support, emotional understanding and love that I received from my family for their continual support, emotional understanding, and love. Their compassion and tolerance are my greatest source of confidence and encouragement in completing my senior thesis.

At the conclusion of this thesis, I would want to thank everyone who made it feasible and pleasurable for me, as well as those who provided me with assistance.

LUNG CANCER DETECTION ON CT SCAN IMAGES WITH DEEP LEARNING METHODS: SUGENO FUZZY INTEGRAL-BASED CNN ENSEMBLE

ABSTRACT

Currently available deep learning studies on lung cancer detection are not ideal. Existing approaches to lung cancer detection are primarily focused on a single model or multiple models using probability averaging. All deep learning models have limitations in image processing, and the possibility of inaccurate results that occur due to those limitations will impede the diagnosis. This issue must be rectified promptly as lung cancer causes the highest mortality rate among all cancers. With the intention of saving more life, this study proposes the use of Sugeno Ensemble approach in combination with four Convolutional Neural Networks (CNN), VGG11, SqueezeNet, GoogLeNet, and Wide ResNet-50-2 to achieve a higher accuracy for lung cancer detection to allow medical practitioners to treat the patients without delay. This study analyses the accuracy of utilising Sugeno fuzzy logic techniques on CT-scan images of lung cancer to see if it outperforms existing approaches and investigate the shortcomings of previous studies. This study looks at various cancer morphologies from various images of patients. The confusion matrix is then used to evaluate the framework's performance. The final accuracy result attained is 98.47%. This study discovered that combining the Sugeno ensemble approach with the four CNN models improves the overall performance of lung cancer detection. This findings could aid medical practitioners in designing tools to identify patients suffering from lung cancer and provide implications in artificial intelligence for medical practitioners and the medical industry.

PENGESANAN KANSER PARU PADA CT SCAN GAMBAR DENGAN KAEDAH PEMBELAJARAN DALAM: ENSEMBLE CNN BERASASKAN INTEGRAL SUGENO FUZZY

ABSTRAK

Kajian pembelajaran mendalam yang tersedia pada masa ini mengenai pengesanan kanser paru-paru masih tidak sesuai. Pendekatan sedia ada untuk pengesanan kanser paru-paru tertumpu terutamanya pada satu model atau berbilang model menggunakan purata kebarangkalian. Semua model pembelajaran mendalam mempunyai had dalam pemprosesan imej, dan kemungkinan keputusan tidak tepat yang berlaku disebabkan oleh had tersebut akan menghalang diagnosis. Isu ini mesti dibetulkan segera kerana kanser paru-paru menyebabkan kadar kematian tertinggi di kalangan semua kanser. Kajian ini mencadangkan penggunaan pendekatan Sugeno Ensemble dalam kombinasi dengan empat Convolutional Neural Networks (CNN), VGG11, SqueezeNet, GoogLeNet, dan Wide ResNet-50-2 untuk mencapai ketepatan yang lebih tinggi untuk pengesanan kanser paru-paru. untuk membolehkan pengamal perubatan merawat pesakit tanpa berlengah-lengah. Ketepatan penggunaan teknik logik kabur Sugeno pada imej CT-scan kanser paru-paru untuk mendapati cara ini merupakan cara yang lebih berkesan daripada cara dahulu. Matriks kekeliruan kemudiannya digunakan untuk menilai prestasi rangka kerja. Keputusan ketepatan akhir yang dicapai ialah 98.47%. Kajian ini mendapati bahawa menggabungkan pendekatan ensemble Sugeno dengan empat model CNN meningkatkan prestasi keseluruhan pengesanan kanser paru-paru. Kertas ini boleh digunakan untuk mencipta alat untuk membantu pengamal perubatan mengenal pasti pesakit yang menghidap kanser paru-paru. Penemuan ini memberikan implikasi dalam kecerdasan buatan untuk pengamal perubatan dan industri perubatan.

TABLE OF CONTENTS

LIST OF TABLES

LIST OF FIGURES

LIST OF ABBREVIATIONS

Abbreviations	
CNN	Convolutional Neural Network
CT	Computed Tomography
FN	False Negative
FP	False Positive
IQ-OTH/NCCD	Iraq-Oncology Teaching Hospital/National Center for Cancer Diseases
LIDC-IDRI	The Lung Image Database Consortium and Image Database Resource Initiative
TN	True Negative
TP	True Positive
UMT	Universiti Malaysia Terengganu

LIST OF APPENDICES

CHAPTER 1

INTRODUCTION

1.1 Introduction

The medical industry is blooming to become more modern in this new era of globalisation, the medical sector is developing, and the sectors have been competing with one another and working together to discover a solution for many ailments that are fatal. In order to find a treatment for the fatal disease, it is necessary to first be able to detect the presence of the disease, specifically cancerous tumours and the formation of malignant cells. These are often missed by the naked eye due to the fact that some of the pictures are obscure and perplexing. Cancer is one of the diseases that is most frequently disregarded. Cancer can arise in the human body in a variety of different ways, such as when an error occurs in the deoxyribonucleic acid (DNA) of cells, causing those cells to proliferate irregularly or through parental inheritance. Both of these scenarios are potential causes of cancer. In most cases, our bodies have the capacity to eradicate cells that have defective DNA before those cells may develop into cancerous tumours. This power, however, decreases with age, and as a result, cells that contain mistakes are able to proliferate, which ultimately results in the development of cancer (National Cancer Institute, 2021). Metastasis refers to the process in which cancer cells go from one part of the body to another, such as from the lungs to the kidney. Due to the fact that it interferes with the body's natural functions and leads to organ dysfunction, the metastatic process is the leading cause of mortality in cases involving cancer (Cancer.Net, 2019).

Cancer is one of the diseases that, when it progresses to a later stage, almost always lethal and fatal and is still not treatable; this is especially true with lung cancer. There are around 2.21 million new instances of lung cancer diagnosed each year, making it one of the most frequent forms of the disease. Lung cancer is also the main cause of death, with an expected incidence of 1.80 million fatalities recorded in the year 2020. (World Health Organization, 2021).

Table 1-1 Type of Cancer

Type of Common Cancer	**New Cases in 2020 (million)**	**New Cases in 2020 (%)**
Breast Cancer	2.26	22.4
Lung Cancer	2.21	21.8
Colon and Rectum	1.93	19.1
Prostate	1.41	14.0
Skin	1.20	11.9
Stomach	1.09	10.8

Source: Cancer, World Health Organization (2021)

Table 1-2 Type of Cancer Death

Type of Cancer Death	**New Cases in 2020 (million)**	**New Death in 2020 (%)**
Lung Cancer	1.80	35.9
Colon and Rectum	0.93	18.5
Liver	0.83	16.5
Stomach	0.77	15.3
Breast	0.69	13.7

Source: Cancer, World Health Organization (2021)

Tobacco use, which is responsible for around 80% of lung cancer cases worldwide, is the primary contributor to the alarmingly increasing number of newly diagnosed instances of lung cancer. It is estimated that 25% of people who have been diagnosed with lung cancer have not experienced any symptoms whatsoever, and the remaining 75% of people are only diagnosed with it when some kind of symptoms began to develop. In most cases, the symptoms won't appear until the malignant cell has already moved to other areas of the body, at which point it will have already produced a disruption in the human blood, hormones, and even other bodily systems (Singh, 2018).

Chest X-rays, which can provide images of the lungs, are the method that can be used to identify lung cancer. The detection of larger tumours is typically possible with this method, but a diagnosis of lung cancer is commonly overlooked. This is because X-ray scans can only be viewed in various shades of grey, and it was necessary to have radiologists analyse any anomalies found. Additionally, the images that were accessible did not have a very high quality. Because the image is not good enough, it is easy to miss small tumours, and these tumours are typically only discovered in their advanced stages after they have become enormous. Sometimes, radiologists were unable to discern between substances with comparable densities, such as pus and water, because occluded images, which are also overlapping images of the tumour on the X-rays, caused them to be unable to see the differences between the two. Because tumours are typically concealed within pus as well, it is impossible to detect them using standard two-dimensional X-ray pictures. This is due to the fact that tumours have three dimensions. Because of this, the use of X-ray pictures to screen for lung cancer has some restrictions, and as a result, a different approach is required to screen for lung cancer (Douglas, 2020).

Computed tomography has largely taken the place of X-rays in medical practise (CT). The computed tomography (CT) scan machine is a massive piece of equipment that is typically found in hospitals or research laboratories. It has a doughnut-shaped tube that allows the X-ray to be rotated throughout the patient's body in order to acquire images that can be taken from any angle around the patient. Therefore, a CT scan is a kind of digital imaging method that generates a three-dimensional image from a large number of two-dimensional images. This is more advantageous than using regular X-rays because it eliminates the problem of overlapping structures as the CT images produced are taken through various axis of the scanner and have multiple detectors to increase the resolution of the images. In addition, CT scans are faster than regular X-rays (Corno, 2009). This is because it can identify both known and unknown lung cancer tumours, this test is the superior option for screening for lung cancer. It is recommended that it be utilised (Amal et al, 2014). In addition, contemporary CT scanners have the ability to produce an image with a "low noise," which helps to reduce the "noise" that would otherwise damage the overall images when the radiologist or other software used to process the images was doing so (Kevin, 2018).

In addition to this, deep learning has demonstrated its value in a variety of scientific fields. Several distinct approaches to deep learning each serve a unique function within the context of this significantly developed and complex field. This is because a large number of researchers are interested in doing studies on deep learning, and these studies are also being evaluated by a large number of other academics to demonstrate the utility of deep learning (Litgens, 2017). In the past five years, an immense number of scholars had made their way into understanding what this field had to offer. Deep learning for medical applications is also an active area of research by other researchers (Pandey, 2021). As a result of this, deep learning is now being rapidly implemented in various fields of the medical field, most notably in the field of medical imaging. This is because the field of medical imaging is an essential sector that has the potential to usher in a new era in the sector of the medical field (Maier, 2019).

There have been many studies conducted in the past by a wide range of researchers with the goal of increasing the detection rate closer to one hundred percent. However, there are still certain limitations to the previously conducted research, which are a challenge for a large number of researchers. For instance, the watershed segmentation that is used to diagnose lung cancer (Makaju et al, 2018) has the issue of producing an over-segmentation image due to its inability to penetrate far enough into the tissue (Karthikeyan, 2012). Therefore, innovative approaches are required to find a solution to this issue.

However, each method has its own advantages and disadvantages; hence, a variety of ways will be utilised, and these methods will be combined using an ensemble method. Methods such as VGG-11 (Simonyan et al., 2014), SqueezeNet v1.1 (Iandola et al., 2016), GoogLeNet (Szegedy et al., 2015), and Wide ResNet-50-2 will be utilised in this study (Zagoruyko et al, 2016).

Furthermore, this research suggested employing the Ensemble Learning methodology, which is the Sugeno Fuzzy Integral, for the fusion and the generation of the final scores to forecast the images of various deep learning methods in order to acquire an accurate reading for the photos. Although in most cases, only straightforward fusion strategies are used when combining different models. However, these straightforward fusion approaches do not take into account the various models. For instance, the weightage of a specific model could not be quite as accurate, but the other models might be able to recognise the product with pinpoint accuracy. As a result, the Sugeno fuzzy integral can be utilised to make up for the deficiencies caused by the simple fusion procedure. They are able to offer the weightage of each input to the model by taking into consideration the confidence of each prediction. This is in contrast to traditional fusion approaches, which employ fixed weights (Kundu et al., 2021).

The dataset that will be used is the next topic to be covered once we have completed our discussion on the approach that will be utilised in this study. The IQ-OTH/NCCD - Lung Cancer Dataset is going to be utilised as the dataset of choice. The Iraq-Oncology Teaching Hospital and the National Center for Cancer Diseases will be the source of the dataset that will be collected, and the PNG file format will be used for the picture extension.

Last but not least, this research will follow the general approach of lung cancer detection, which is shown in Figure 1-1, which consists of 5 general steps. The first stage is the image acquisition where the images are taken from the dataset above. The second step will be image pre-processing and segmentation which the research will focus on various image processing approaches to enhance the image for easier interpretation and feature extraction. The third step is feature extraction, which the images processed in the second step will have its' feature to be extracted to be used for the classification in the fourth step. The fourth step is to classify the image whether it possesses the desired characteristics. The fifth step, which is also the last step will be used to evaluate the implementation of each image processing approach for lung cancer detection (Singh & Gupta, 2018).

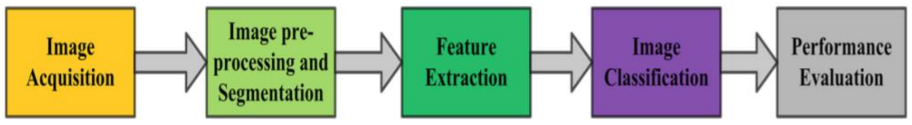


Figure 1-1 Five basic steps for detection of lung cancer in humans

1.2 Research Background

The lung cancer had caused many families to be devastated as this is a type of fatal illness and many humans had passed away due to this illness, the symptoms of this disease also will show only when the patient is in a dangerous phase. Lung cancer can be detected with CT Scan images that will be usually determined by the radiologist. However, radiologists may sometimes fail to recognise the symptoms as the cancerous tumour may be too small or is unrecognisable until the radiologist has neglected it. Therefore, this research is carried out to understand the benefit of image processing in deep learning towards CT Scan images as this can serve as an assistance to the radiologist to make a more accurate detection of lung cancer.

In this research, the method to improve the efficiency of lung cancer detection is using the Convolutional Neural Network (CNN) by using a fuzzy integral based ensemble. The use of four different architectures is applied to ensure that the different characteristics and features of lung cancer can be detected successfully. Figure 1-2 shows the workflow of the proposed methods. The corresponding nature of different methods is compared and the divergences from the decision scores of the model is then fused together using the Sugeno Fuzzy Integral to have a more accurate score as this ensemble method can condition the weight of each of the different CNN methods used.

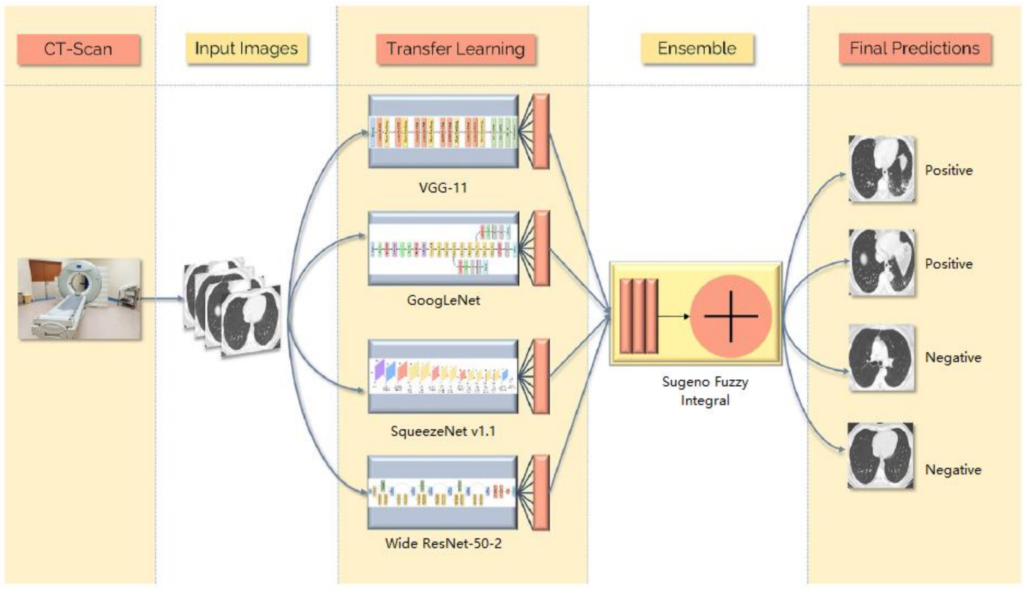


Figure 1-2 Overall workflow of the proposed approach

The fuzzy logic is also relatively new to the medical industry. Fuzzy logic can be used whenever there is an uncertainty for the weightage of various factors of methods proposed. Therefore, the usefulness and ways of fuzzy logic work is also important and needs to be addressed and understood.

There are many research available on the internet about lung cancer, many of which provide an accurate result for predicting lung cancer. However, there is one critical issue, which is that the methods proposed by many different researchers only use a single classification or a single method to check the presence of lung cancer, which is not particularly useful as the shape and characteristics of cancer may differ by person (Morozov, 2021). There are also some other researchers that use probability averaging when various methods are used together, this will be less accurate as either one of the inaccurate readings will affect the entire percentage. Thus, this fuzzy integral-based ensemble is able to provide adaptive priority to the methods available on the fly.

1.3 Problem Statement

i. What is the accuracy of Sugeno fuzzy ensemble combined with CNN Models to assist in CT scan of lung cancer?
ii. How does Sugeno fuzzy ensemble combined with CNN Models compare to existing research?
iii. How to make Sugeno fuzzy ensemble combined with CNN Models more accessible for the study?

1.4 Research Objectives

The objectives of the research are:

i. To study the accuracy of Sugeno fuzzy ensemble combined with CNN models to assist in lung cancer detection.
ii. To evaluate whether Sugeno fuzzy ensemble combined with CNN models outperform existing methods.
iii. To develop a prototype of lung cancer detection using Sugeno fuzzy ensemble combined with CNN models.

1.5 Research Scope

This research will only focus on lung cancer because the number of lung cancer is the primary cause for death in 2020, with an estimated number of 1.80 million deaths recorded in 2020 (World Health Organization, 2020). This is because smoking will cause lung cancer in men and women. Thus, if smoking rate is not decreased, the number of humans with lung cancer will increase.

Besides that, the CT scans of lungs are taken from The Cancer Imaging Archive Public Access. This is especially useful as there are a large number of images available, which are 251,135 images.

Furthermore, the research will only focus on the use of Fuzzy Integral-Based CNN Ensemble to compare with existing methods. This is to determine whether this method is able to replace existing methods to help radiologists to detect the presence of lung cancer.

Finally, the coding for this research will be done using Python. Python takes the lead compared to other programming languages for deep learning with more than 60% of deep learning developers using it for their research and development. This is because Python has many useful libraries to make work easy and enable machines to learn (Banerjee, 2021).

1.6 Thesis Organization

Chapter 1 describes the background of this research, problem statement, problem significance, the objective of this research, scope of the research, organization of this thesis.

Chapter 2 explains the literature review of this research, which includes the introduction to machine learning, deep learning, image processing and convolutional neural network. This chapter also includes the review of papers that currently exist.

Chapter 3 elaborates the methodology of this research. It describes the type of data that are collected, the method of data pre-processing, transfer learning, the Sugeno ensemble and the method used for determining the performance of the approach.

Chapter 4 clarifies the research implementation, which includes the data sample as well as the interface of the proposed approach and the way for the study to be initiated.

Chapter 5 narrates the results and discussion. This is where the results are evaluated to understand the performance of the proposed approach.

Chapter 6 discusses the conclusion as well as the contributions of the research and the fulfilment of the research objectives.

CHAPTER 2

LITERATURE REVIEW

2.1 Introduction

This chapter of literature review provides a detailed discussion on deep learning, and about the use of existing deep learning methods for lung cancer detection.

2.2 Machine Learning

The birth of machine learning can be originated from the seventeenth century to mimic the human ability to do mathematical addition and subtraction by Pascal and Leibniz (Ifrah, 2001).

In modern history, the first breakthrough of machine learning started in 1950 by Alan Turing. (Naeini & Prindle, 2018). Alan Turing proposed his Turing machine for the world (Turing, 1950). After a few years, the concept of machine learning was coined by Arthur Samuel in 1959 (IBM, 2020). He explained that machine learning is the field of study which provides computational devices to discover without obviously obviously being programmed and provide insight on the capabilities of computer learning to engage in checkers (Samuel, 1959). Besides that, an architecture on early neural networks is developed (Rosenblatt, 1958).

Machine learning then started to boom in the 1960s, some of the machine learning research were then published by various authors, such as research about learning recognition by Bongard in 1961, Braverman in 1962 and research about pattern recognition by Widrow in 1964 (Fradkov, 2020). Numerous new modern machine learning methods were is also developed in the 1980s and 1990s, such as the inception of decision trees (Quinlann, 1986) and discovery of support vector machines (Cortes & Vapnik, 1995).

Today, the more widely known definition of machine learning is the branch of computational algorithms which are intended to mimic the intelligence of a human by learning and improve through experience without the algorithm being specifically programmed (El & Murphy, 2015). Thus, they can optimize the performance by learning and looking for similar patterns and other information to make a prediction or recommendations (Mining, 2020). Machine learning has 4 categories, which includes supervised learning, unsupervised learning, semi-supervised learning and reinforcement learning (Bertolini et al., 2021).

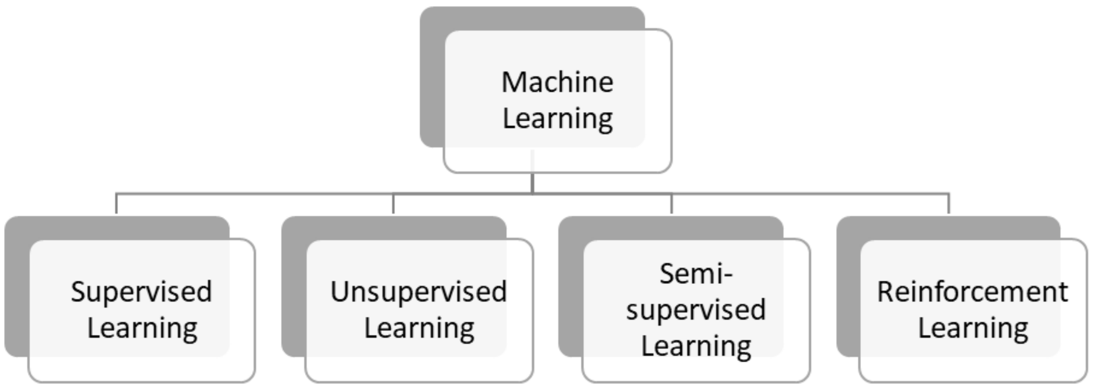


Figure 2-1 Categories of Machine Learning

2.2.1 Supervised Learning

Supervised machine learning is where the machine learning technique learns from a set of labelled data and uses its algorithm to understand the data which is then learned to seek a similar pattern for making an accurate prediction based on the labelled data given (Bertolini et al., 2021). The detriment of this technique is that for the success of the technique requires massive human effort to accomplish (Zhai et al., 2020). There are various machine learning methods that are supervised, which includes neural networks and decision trees (Xu & Jackson, 2019).

2.2.2 Unsupervised Learning

Unsupervised machine learning is the machine learning technique that learns from unlabelled data, which then recognises the structure and pattern of data based on the similarities in pattern (Zhai et al., 2020). Unlike supervised learning, this technique does not require much human efforts which is one of its benefits. Example of unsupervised learning is latent Dirichlet allocation (Bertolini et al., 2021).

2.2.3 Semi-supervised Learning

Semi-supervised machine learning is the machine learning technique in which both labelled and unlabelled data is provided, this technique is usually used for the examination of pattern of labelled data to find similarities and predict patterns in the unlabelled counterparts (Zhai et al., 2020). This method is good as it can overcome the disadvantage of lack of accuracy of unsupervised learning and time and cost for the preparation of supervised learning (Zhai et al., 2020). Examples of semi-supervised learning are various clustering techniques (Bertolini et al., 2021).

2.2.4 Reinforcement Learning

Reinforcement machine learning is the machine learning technique in which no predefined data is needed to be given. The algorithm makes decision according to its environment to learn from the errors and rewards (Treloar et al, 2020). This technique is good to use for data which will change with different timescale as there is no consistent pattern (Neftci & Averbeck, 2019). Example of reinforcement learning is a stacked hierarchical attention network (Xu et al, 2020).

2.3 Deep Learning

Deep learning is a deeper understanding of machine learning technique that concentrates in the development of neural network models for the ability of the creation of precise data-driven decisions (Guo et al., 2016). The interesting aspect of deep learning will mimic the brain of the human in the terms of observation, analysis, learning and making decisions for problems that are complicated and complex (Najafabadi et al, 2015).

Deep learning is made up of layers of nodes, which is similar to the human brain which has neurons. All the individual nodes will be connected to its adjacent layer and will pass the signal to the next layer, the deep of the layer depends on the number of nodes it has. The signals will travel between the nodes to provide weights as it is important for heavier nodes to have more effect on the next layers. Afterwards, when the nodes reach the final layer, the final layer nodes will compile all the input to produce a decision (Buckner & Cameron, 2019).

Deep learning can save the time of humans as human intervention is reduced for feature design. It can learn the features which are beneficial for the task from the raw data instead of requiring large human effort for the feature identification. Datasets which are large and huge are better to train deep learning algorithms. Through the increase of data provided, it is proven that this deep learning will get more effective in the identification of features of the data (Kelleher, 2019).

There are currently many applications in the real world using deep learning, some of the fields that are using this deep learning methods are chatbots in customer service (Dhyani & Kumar, 2021), medical research (Wang et al., 2021) and computer vision (Arya et al., 2021).

2.4 Image Processing

Image processing is the handling of a file whose properties are an image using a computing device (da Silva & Mendonça, 2005). Image processing currently possesses many applications and an important feature to replace traditional methods. Traditionally, an expert is employed to detect and check for the image or real-world object to detect the condition of the object and classify it manually. Additionally, the price may be high for consultation as it is expensive and not many sectors have enough budget to hire the expert, the process of the traditional method is time consuming as well (Suganya et al, 2019).

Image processing can be brought back to the early 1920s, which was first used by the newspaper industries to reduce the time taken for transporting images (Shivajirao, 2011). Afterwards, image processing had undergone various improvement techniques and during 1964, it had one of the most notable moments, which was to improve the image of moon taken by the Ranger 7 space probe (Gonzalez et al, 2009). In 1972, computer assisted tomography was invented by Godfrey Newbold Hounsfield and they received the Nobel Prize in 1979 after various tests were conducted (Petrik, 2006).

In today's world, image processing has gained popularities and interest of various researchers and industry sector as it focuses on two of the main key areas:

- Enhancement of image data for human understanding and analysis.
- Processing of image data for computing assessment

There are currently many uses for image processing for different sectors and areas and are among the hot topics for using image processing together with deep learning. Various fields have started to apply deep learning in various tasks, such as medical science (Treloar et al, 2020), robot vision (Cheng et al, 2020), farming (Suganya et al, 2019), remote sensing (Jung et al, 2021), transmission and encoding (Alhayani et al, 2021).

2.5 Convolutional Neural Network

Convolutional Neural Network is a type of deep learning architecture which is used in image processing. This network is able to reduce the initial complexity of traditional neural networks, with local connections, weight sharing, and other characteristics and have been well used in various fields, which includes computer vision and natural language processing (Chen et al, 2021).

This architecture is inspired by the visual cortex of animals, and thus this architecture is known for their ability to extract strong spatially strong correlation to possess a high-order feature extraction effect from the unprocessed data obtained (Patterson & Gibson, 2017). As a result, his architecture has the advantage of no manual labour of feature engineering, which are both feature extraction and feature selection. These feature engineering that are usually selected and picked by domain engineers are not required (Lecun et al., 2015).

Convolutional Neural Network comprises of various layers, which are convolutional layer, pooling layer and fully connected layer. Each of the layers are tasked with different work for this deep learning architecture to perform well.

The convolutional layers (filter or kernel) is also known as the feature extractor. The kernels are applied on the data of the image to produce a feature map. There are various layers that are tasked to capture different features available. For instance, the first layer is tasked to detect the presence of edges and the second layer is for the detection of combination of edges (Silva, 2019). The next thing that is important to convolutional layers are strides and padding. Strides are the number of pixels in which the kernel slides over the matrix. Padding will be the assistance when data in the kernel is not the same size as the input matrix (Srinivas et al., 2016).

Pooling layer is tasked in the reduction of the dimensionality of the feature maps, which is through reducing the height and width of the feature map while preserving the depth of the image. This step is important as it can help to decrease computation needed for data processing while still capable of extracting deep features needed in the feature map (Dao, 2020).

Finally, the fully connected layer is the layer for classification. The matrix will be flattened and turned into a column vector to go through a fully connected layer. The fully connected layer then passed through activation functions, ReLu and Softmax. The activation layer has an N-dimensional vector, which determines the classes of which the architecture needs to decide (Dao, 2020).

2.5.1 AlexNet

AlexNet is a type of CNN model in image processing of the category of image classifying and was developed in 2012 (Krizhevsky et. al, 2012). This model was announced during the ImageNet Large Scale Visual Challenge in 2012. AlexNet also provides a significant impact on the advancement of CNN (Chen et al, 2021). It is an amazing model as it was trained with over one million images and possesses the capability to recognise 1000 different objects (Togaçar et al., 2020). Table 2-1 shows the architecture of AlexNet.

Table 2-1 Architecture of AlexNet

Layer		**Feature Map**	**Size**	**Filter Size**	**Stride**	**Activation**
Input	Image	1	227 x 227 x 3	11 x 11	4	relu
1	Convolution	96	55 x 55 x 96	3 x 3	2	relu
	Max Pooling	96	27 x 27 x 96	5 x 5	1	relu
2	Convolution	256	27 x 27 x 256	3 x 3	2	relu
	Max Pooling	256	13 x 13 x 256	3 x 3	1	relu
3	Convolution	384	13 x 13 x 384	3 x 3	1	relu
4	Convolution	384	13 x 13 x 384	3 x 3	1	relu
5	Convolution	256	13 x 13 x 256	3 x 3	2	relu
	Max Pooling	256	6 x 6 x 256	-	-	relu
6	Fully-Connected	-	4096	-	-	relu
7	Fully-Connected	-	4096	-	-	relu
8	Fully-Connected	-	1000	-	-	relu
Output	Softmax	-	1000	-	-	Softmax

There is various research that uses the AlexNet model. According to Almas, the performance of AlexNet is amazing as it has a high accuracy and learning rate while training time is low on lung cancer (Almas et al, 2019). Togaçar utilised this model and found out that it extracts a total of 1000 features from the Cancer Imaging Archive dataset (Togaçar et al., 2020).

According to Aditi, this AlexNet is effective for the detection of breast cancer detection using the image provided by the mammogram, which X-ray used for breast cancer detection, and through this model, it is able to achieve an accuracy of 92.00%, sensitivity of 81.50% and specificity of 90.83%. However, it has a lack in the ability for detection for images with low contrast in the surrounding area alongside with a complex nature (Aditi et al, 2021).

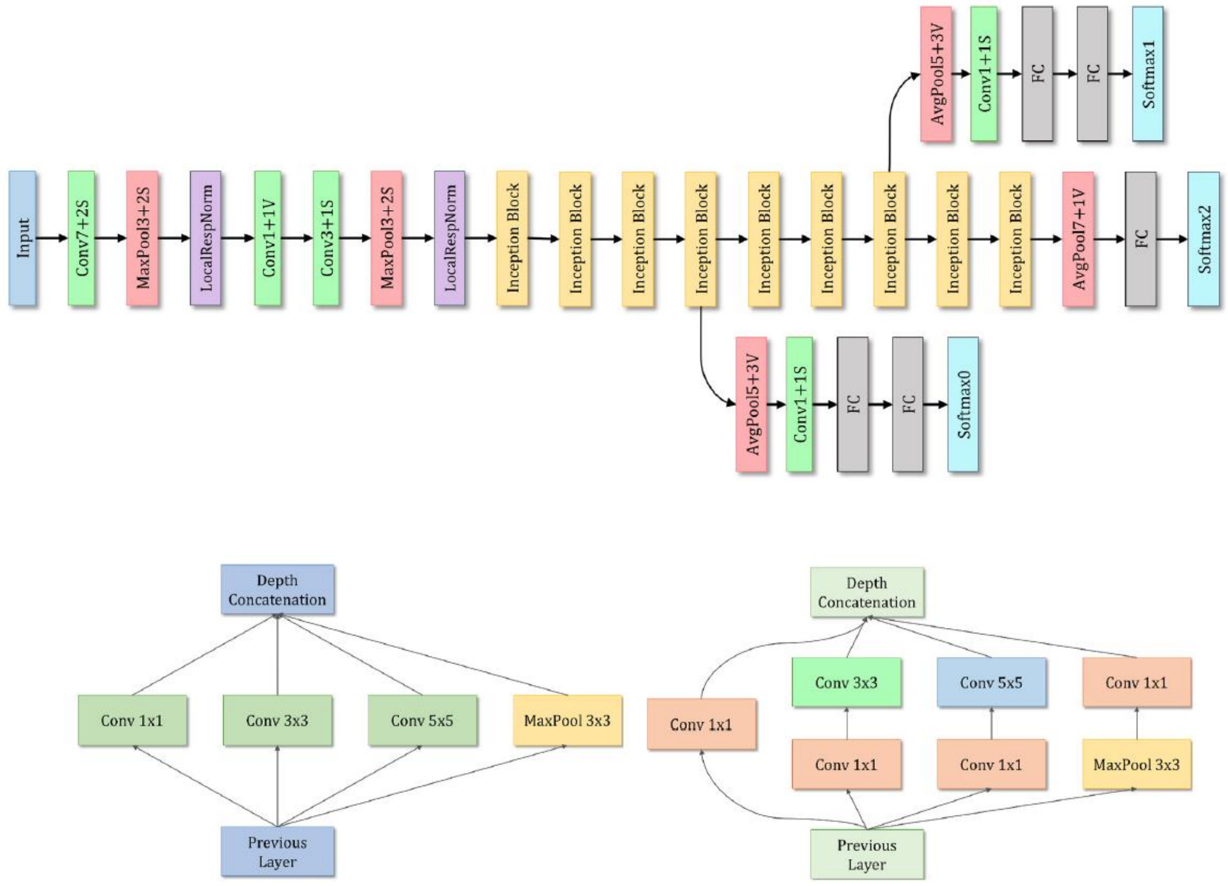
According to Zhang, this AlexNet is also effective in the detection of breast cancer as it can improve the detection rate and achieves a rate of 87.69% in accuracy. However, it still pale in comparison when compared with GoogLeNet, which achieve accuracy of 91.18% due to having a less advanced feature extraction capabilities.

2.5.2 GoogLeNet

GoogLeNet is a type of CNN model in image processing and is used for image analysis and was developed in 2015 (Szegedy et al, 2015). This model was announced and participated in competitive vision challenges, Imagenet 2014 and obtained a distinguished result of 93.33% accuracy in the classifying of daily objects (Russakovsky et al, 2015).

GoogLeNet is used to solve the problems which large networks face, which are overfitting through the Inception module. This leverages the feature detection at different scales through the convolution and reduces the computational costs of training the models through dimensional reduction (Szegedy et al, 2015). GoogLeNet consists of 22 hidden layers, thus the depth of this neural network is larger than AlexNet (Sajja et al, 2019), which makes it more effective in image classification.

There is various research pertaining to lung cancer that use this model. According to Sajja, this model can reduce the overfitting during the learning phase and achieve a good accuracy (Sajja et al, 2019). Al-Huseiny & Sajit also mentions that GoogLeNet is able to accommodate the data for the sensitivity and specificity to further prove its reliability (Al-Huseiny & Sajit, 2020).



(a) Naive inception block

(b) Inception block with dimensionality reduction

Figure 2-2 Architecture of GoogLeNet

2.5.3 LeNet

LeNet is a type of CNN model in image processing and is used for image analysis and was developed in 1988 and had undergone improvements until 1998. (Lecun et al, 1998). LeNet possesses a 7-level convolution network in which 2 of them are convolution layers, 2 are sub-sampling layers, 2 are FC layers followed by an output layer with Tanh as an activation function and the back propagation paradigm. (Thakur et al, 2020).

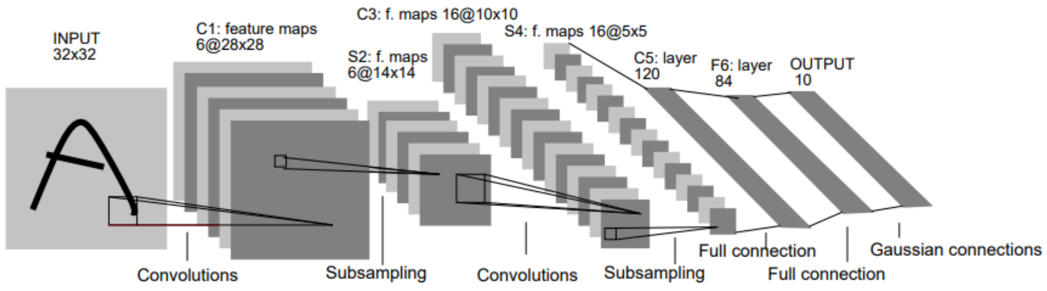


Figure 2-3 Workflow of LeNet

There are various studies that use this model. According to Togaçar, it is the worst performing model when compared to AlexNet and VGG-16 (Togaçar et al., 2020). This is due to it having the least level of convolution network which hinders its accuracy.

2.5.4 VGG-11

VGGNet is a type of CNN model in image processing and is used for image analysis and was developed in 2014 (Simonyan & Sizzerman, 2014). VGGNet consists of three different architectures with varying convolution layers. Starting with VGG-11 which contained 8 layers, VGG-16 contains 13 layers, and VGG-19 contains 16 layers of convolution. These are all followed by a single max-pooling layer and 3 fully connected layers, and the last layer is a Softmax layer for classification. (Simonyan & Sizzerman, 2014).

For a clearer explanation for VGG-11. At the convolution layer, the kernel size is 3 x 3, which is also the size of the filter. Next, the max-pooling layer in which the convolution stride is 1 pixel. Last but not least, there are 3 fully connected layer for VGG-11, which 2 have 4096 channel and the last one has 1000 channels. There is also a hidden layer, which is ReLU. ReLU has several advantages, which are able to be applied on big neural networks as it is able to reduce duration of training and thus reduce computational cost for training. (Simonyan & Sizzerman, 2014).

There is various research that utilises this model. According to Simonyan, this is model is effective as capable for object-detection and takes account of overfitting and is more suitable for large-scale files for image recognition and a better replacement for AlexNet as it reduces the parameter and have a lower training time (Simonyan & Sizzerman, 2014).

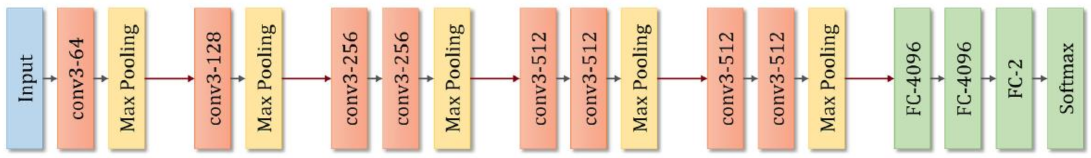


Figure 2-4 Architecture of VGG-11

2.5.5 ResNet

ResNet is a type of CNN model in image processing and is used for image analysis and was developed in 2015 (He et al., 2016). This model won the first place in ILSVRC 2015 classification competition (Izadkhah, 2022). ResNet has 152 layers. This research is conducted as the author wishes to determine network architecture needs to go deeper for a higher accuracy. However, it is realized that increasing in depth is by simply stacking the layers together due to having vanishing gradient issues and having a maximum threshold. This causes saturation and degradation issues.

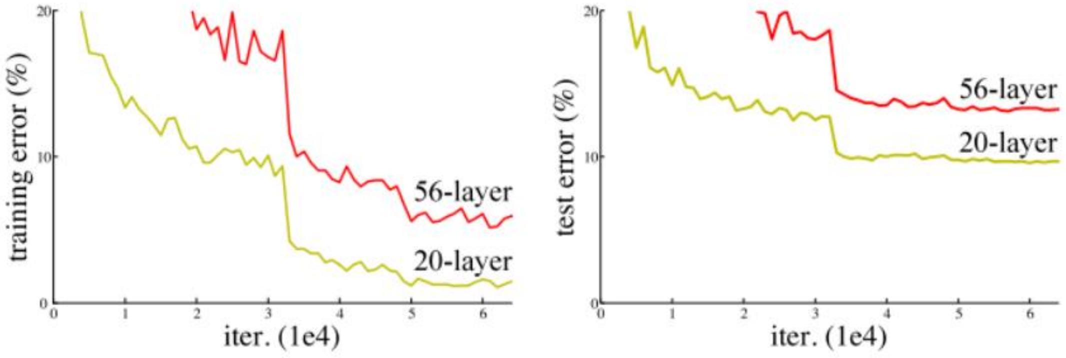
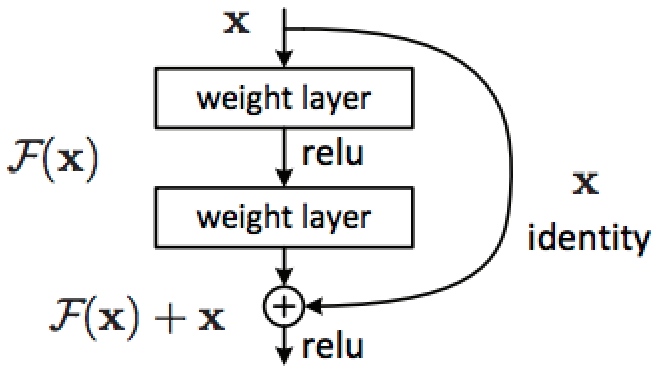


Figure 2-5 The training error by simply stacking layers together

Thus, with ResNet comes the introduction of residual blocks, instead of stacking layers on each other to let all layers learn the mapping, it has a skip connection to allow for reducing the bias.



x
weight layer
relu
weight layer
$\mathcal{F}(\mathbf{x})$
x
identity
$\mathcal{F}(\mathbf{x}) + \mathbf{x}$
relu

Figure 2-6 Residual Block of ResNet

2.5.6 SqueezeNet

SqueezeNet is a type of CNN model in image processing and is used for image analysis and is developed in 2016 (Iandola et al, 2016). This SqueezeNet has 14 layers and is the replacement for AlexNet. It has 50 fewer parameters than AlexNet and 510 fewer parameters after compression but still managed to perform 3 times faster than AlexNet. SqueezeNet consists of a squeeze and expand layer, which is called a fire module. In which the squeeze layer only has 1 x 1 filters, which will be fed into expand layers with a mix of 1 x 1 and 3 x 3 filters.

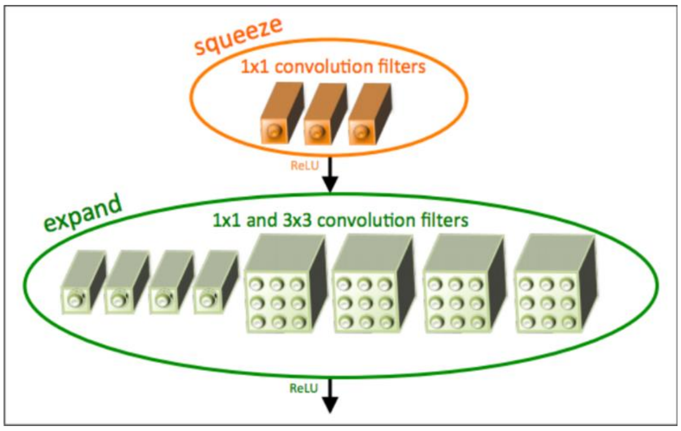


Figure 2-7 Squeeze and Expand Block of SqueezeNet

There are three types of SqueezeNet architecture in which starts with the convolution layer with the follow up by eight fire modules and ending with a convolutional layer. The number of filters increases with the stages, starting with the lowest in the beginning and highest in the end. The max pooling layer with stride of value 2 is also found in a few layers, which are after the first convolution layer, after the third fire module, after the seventh fire module and after the second convolution layer. There are also different SqueezeNet architecture, on the left will be the architecture explained earlier, on the middle will be SqueezeNet that have a simple bypass and last but not least the right side will be the Squeezenet with a complex bypass that is inspired with the use of ResNet. The diagram of the architecture is shown below. Figure 2-8 shows the architecture of SqueezeNet.

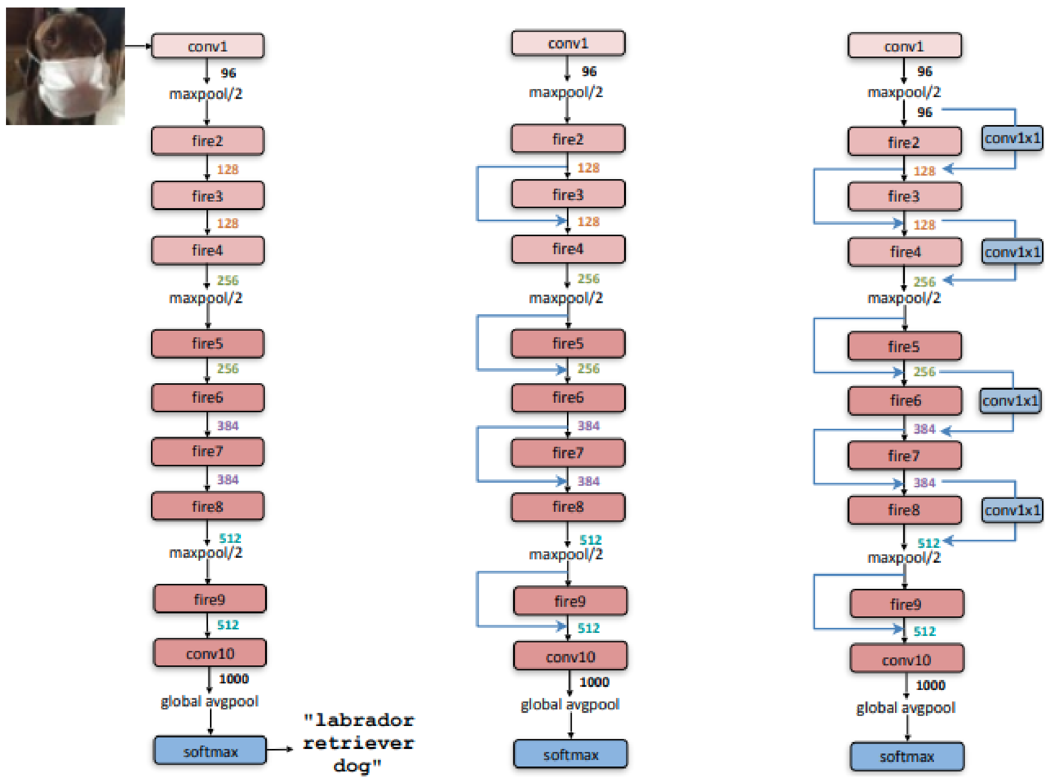


Figure 2-8 Architecture of SqueezeNet

2.6 Comparison of Related Research

Table 2-2 Existing Research Compared

Title	Author/Year	Dataset	CNN Models
Comparison of deep learning convolutional neural network (CNN) architectures for lung cancer classification	Ashhar, S. M., Mokri, S. S., Rahni, A. A., Huddin, A. B., Zulkamain, N., Azri, N.A, Mahaletchumy, T. (2021)	LIDC-IDRI	GoogLeNet, SqueezeNet, ShuffleNet, DenseNet, MobileNetV2
COVID-19 Detection From Lung CT-Scans Using a Fuzzy Integral-Based CNN Ensemble	Kundu, R., Singh, P. K., Mirjalili, S., Sarkar, R. (2021)	Dataset by Soares et al. (2020)	SqueezeNet v1.1, GoogLeNet, WideResNet-50-2 and VGG-11
Diagnosis of lung cancer based on CT scans using CNN	Al-Yasriy, H. F., Al-Huseiny, M. S., Mohsen, F. Y., Khalil, E. A., Hassan, Z. S. (2020)	IQ-OTH/NCCD	AlexNet
Lung Nodule Detection in Computed Tomography Scans Using Deep Learning	Dobko (2019)	LUNA16	VGG-11
Transfer learning with GoogLeNet for detection of lung cancer	Al-Huseiny, M. S.& Sajit, A. S. (2021)	IQ-OTH/NCCD	GoogLeNet

2.6.1 Transfer learning with GoogLeNet for detection of lung cancer

This research is done by AL-Huseiny and Sajit. (2021) and they used convolutional neural networks in this research, in which GoogLeNet is selected as the model for their study.

The authors carried out this research in the hope of seeking enlightenment towards lung cancer detection. For the research, the dataset used was from Iraq-Oncology Teaching Hospital/National Center for Cancer Diseases, in which a total of 1190 images from 110 patients were selected. A short review on this dataset shows that this database has a few flaws, such as the dataset having noise presence in the images.

The research describes the proposed method for this first is through pre-processing steps, which is utilized through Gabor filter (Daugman, 1988). This filter is used for texture analysis, this pre-processing method is able to highlight the area with related texture for a better detection in lung cancer.

Afterwards, GoogLeNet is used as the model to train with the images provided using the training set. This GoogLeNet can learn through the features of this dataset to obtain a high accuracy.

Table 2-3 Result Obtained from Transfer learning with GoogLeNet for detection of lung cancer

Performance Metrics	Performance Value (%)
Sensitivity	95.08
Specificity	93.70
Accuracy	94.38

From this result, we can see that the accuracy attained by GoogLeNet is able to obtain a result of 94.38%. This architecture used can obtain a high accuracy, this is because suitable pre-processing is able to make the dataset uniform and easier for the architecture to train, there is a result given by the author in which only 70% is obtained when pre-processing is not used.

2.6.2 Diagnosis of Lung Cancer Based on CT-Scans using CNN

This research is done by Al-Yasriy et al (2020) and they use convolutional neural networks in this research, in which AlexNet is selected as the model for their study.

The authors carried out this research in the hope of seeking enlightenment towards lung cancer detection. For the research, the dataset used was the same as the previous research, which is from Iraq-Oncology Teaching Hospital/National Center for Cancer Diseases, in which a total of 1190 images from 110 patients were selected. The dataset is very diversified as the cases involved are patients from different gender and age.

The proposed method suggests using CNN to be applied to detect lung cancer from CT Scan images as it is able to be used for processing images and the model that will be used is AlexNet. AlexNet is effective in detecting this dangerous cancer due to its ability in extracting features contained in images with a mixture of linear “convolution operation” and nonlinear processes “activation function”.

The dataset is separated into two groups, which is 70% for training purposes and 30% for testing purposes. Afterwards, the result is obtained at the end of the training and is shown below.

Table 2-4 Result Obtained from Diagnosis of Lung Cancer Based on CT-Scans using CNN

Performance Metrics	**Performance Value (%)**
Sensitivity	95.71
Specificity	95.00
Precision	97.10
F1 Score	96.40
Accuracy	93.55

From this result, we can see that the accuracy by AlexNet is able to obtain a result of 93.55%. Although this architecture can obtain a good result, there are still questions whether this architecture is able to obtain similar result for reliability and to solve the problem of overfitting, which is often an issue with this architecture

2.6.3 Lung Nodule Detection in Computed Tomography Scans Using Deep Learning

This research is done by Dobko (2019), and she used convolutional neural networks in this research, in which VGG-11 is selected as the model for the study.

The author carries out this research in the hope of producing a high sensitivity method to detect lung cancer in the lungs with CT images. For this research, the dataset used is the LUNA16, which extracts 888 images from The Lung Image Database Consortium and Image Database Resource Initiative (LIDC-IDRI) dataset. 65% of the dataset are divided for training purposes while the remaining 35% will be used for testing.

For pre-processing purposes, random rotation by 90 degrees of the image is initialized and the image is transposed by rows and columns swap, alongside with the brightness and contrast is adjusted and image is scaled to be uniform.

The dataset is then trained using VGG-11, this model is selected as the model for this research as the model is frequently used for feature extraction of various datasets and provide a good result.

Table 2-5 Result Obtained from Lung Nodule Detection in Computed Tomography Scans Using Deep Learning

Performance Metrics	Performance Value (%)
Sensitivity	82.70
Accuracy	96.00

From this result, we can see that the accuracy by VGG-11 is able to obtain a result of 96.00%. This architecture is suitable to detect the presence of lung cancer and various new works can be used to detect the usefulness of this architecture such as using images with different sizes to test the model and change the architecture with deeper architecture, such as ResNet.

2.6.4 Comparison of Deep Learning Convolutional Neural Network (CNN) Architectures for CT Lung Cancer Classification

This research is performed by Ashhar et al (2021) and they used various convolutional neural networks for this research, which are GoogleNet, SqueezeNet, ShuffleNet, DenseNet, MobileNetV2.

The authors carry out this research in the hope of detecting lung cancer as lung cancer is almost unable to be noticed in the early stages. For this research, the data used is the LIDC-IDRI. There are a total of 1646 images which are selected from the dataset. The images are selected and split into 70% for training and 30% for testing.

For the pre-processing process, the images will be resized into 244 x 244 and saved as a PNG file. Afterwards, the dataset will undergo training from GoogLeNet, SqueezeNet, ShuffleNet, DenseNet and MobileNetV2.

Afterwards, the analysis of the dataset is performed to determine the performance of each dataset.

Table 2-6 Result Obtained from Comparison of Deep Learning Convolutional Neural Network (CNN) Architectures for CT Lung Cancer Classification

CNN Architecture	Accuracy	Specificity	Sensitivity
GoogLeNet	94.53	99.06	65.67
SqueezeNet	94.13	99.06	62.69
ShuffleNet	92.91	98.83	55.22
DenseNet	93.52	98.83	59.70
MobileNetV2	92.91	97.65	62.64

From the result, we can see that the highest in accuracy is GoogLeNet, which obtains an accuracy of 94.53%. Therefore, we can say that GoogLeNet is able to obtain a high accuracy when compared with other architectures.

2.6.5 COVID-19 Detection From Lung CT-Scans Using a Fuzzy Integral-Based CNN Ensemble

The research is done by Kundu et al. (2021) and they used various convolutional network models for this research, which are SqueezeNet v1.1, GoogLeNet, WideResNet-50-2 and VGG-11.

The authors carry out the research in the hope of the detection of Covid-19 from using CT scan images. For this research, the collected dataset is provided by Soares et al. (2020). There are a total of 2481 chest images divided into 2 unevenly categories, which is Covid-Positive and Covid-Negative.

The images will undergo 4 pre-trained models as mentioned earlier for transfer learning. VGG-11 is beneficial for accurate classification, SqueezeNet v1.1 is able to achieve similar accuracy with AlexNet with a lesser parameter, GoogLeNet is able to achieve a high accuracy due to having an inception block and last but not least WideResNet-50-2 having a greater width is able to obtain a high accuracy with less time.

After undergoing transfer learning, the result obtained from each of the architectures will undergo the Sugeno ensemble. The Sugeno ensemble uses an adaptive weight for each of the model's confidence in the prediction. Afterwards, the training result of the analysis is recorded.

Table 2-7 Result Obtained from COVID-19 Detection From Lung CT-Scans Using a Fuzzy Integral-Based CNN Ensemble

Performance Metrics	**Performance Value (%)**
Sensitivity	98.93
Accuracy	98.93
Specificity	98.93
F1-Score	98.93

From this result, we can see that the accuracy using this ensemble technique is able to obtain an accuracy of 98.93%. This architecture is useful for the detection of Covid-19 and we can see that the performance value for each of the metrics stays the same due to it having an ensemble method to determine the percentage of Covid-19, the result is very stable and accurate. It will be interesting to determine if the accuracy will remain if it is used to test for lung cancer.

2.7 Conclusion

There are several Convolutional Neural Network models that exist in this system. After various considerations, the models that will be used are VGG-11, GoogLeNet, SqueezeNet v1.1 and WideResNet-50-2. This is because those are models which provide a high percentage of detection rate. VGG-11 is capable for object-detection and takes account of overfitting and is more suitable for large-scale files for image recognition and a better replacement for AlexNet as it reduces the parameter and has a lower training time (Simonyan, 2014). GoogLeNet is used to solve the problems which large networks face, which are overfitting through the Inception module. This leverages the feature detection at different scales through the convolution and reduces the computational costs of training the models through dimensional reduction. SqueezeNet v1.1 is used for image classification and expands the blocks to obtain the deep features and controlling parameter numbers at the same time (Iandola, 2016). Lastly, WideResNet-50-2 is used to optimize the flaws in image, especially in eliminating the gradient vanishing problem and is effective in feature extraction (Zagoruyko 2017).

Besides that, various existing methods that are studied only use one architectural model for the detection of lung cancer and through the result provided, we can see that the results provided by each architecture is different. Thus, it will be interesting to see if there are various types of architecture being used for the detection of lung cancer. Besides that, the ensemble learning method is also less explored in this area of lung cancer detection. Thus, it will be noteworthy for using ensemble method and not just ensemble method with probability averaging and weighted probability averaging which provide priority to the classifiers beforehand, instead fuzzy integral-based ensemble is used to leverage the priority to the classifiers adaptively spontaneously.

CHAPTER 3

RESEARCH METHODOLOGY

3.1 Introduction

In this chapter, we will go over the methodology that was used for this research. This is done so that the efficiency of the suggested method for detecting lung cancer can be evaluated and implemented. The detailing of the technical aspects involved in the implementation of a fuzzy integral-based CNN ensemble for the diagnosis of lung cancer will be the primary emphasis of this chapter. Several distinct procedures for the implementation will be carried out.

This chapter will explain how the structure of this chapter is organised, and how it is broken up into a few key parts.

The first topic that will be covered in this chapter is the data collection process. The term "data collection" refers to both the means by which the necessary dataset required for this study was obtained and the manner in which that dataset was presented.

The second subject that will be discussed is data pre-processing, which entails eliminating unnecessary data in order to obtain a dataset that is consistent throughout.

The third step is known as transfer learning, and it includes the process of extracting and selecting features.

There will be the follow-up, which will be an ensemble of the findings from transfer learning. This will be the follow-up for the ensemble.

The final section will consist of the performance measurement and comparison, which is the comparison and evaluation of the results to establish and identify the suggested method's effectiveness in detecting lung cancer in patients. The purpose of this comparison and evaluation is to determine whether or not the suggested method is effective in detecting lung cancer in patients.

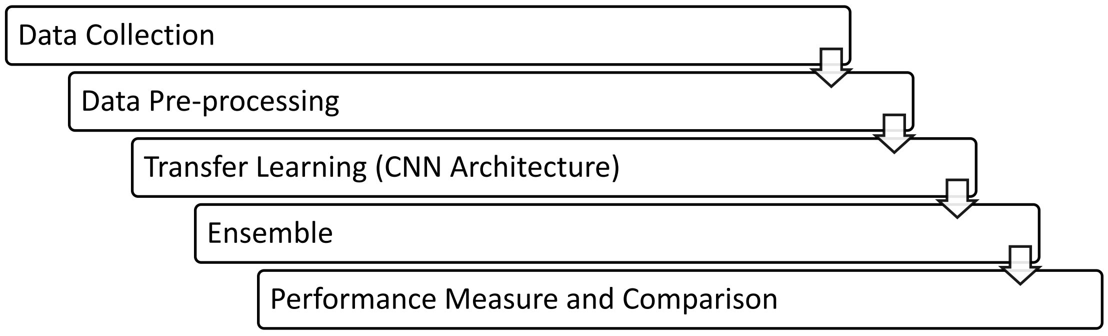

Data Collection
Data Pre-processing
Transfer Learning (CNN Architecture)
Ensemble
Performance Measure and Comparison

Figure 3-1 Framework Of the Study

These sections are separated for better reading and enlightenment purposes. The limitation is that this research will only be putting detection of cancer towards the lungs into account and not cancers in other parts of the living being since this will digress from the original scope of the current research.

3.2 Data Collection

The first thing that needs to be done for the research is to find a credible and extensive dataset of CT-scan images of lung cancer. The photos of lung cancer were obtained from the Lung Cancer Dataset maintained by IQ-OTH/NCCD (Kareem, 2020).

This particular dataset was chosen because the patients come from a diverse range of demographics and span a wide age range and gender distribution.

Therefore, it is reasonable to infer that this dataset is able to accurately diagnose lung cancer in individuals despite their varied backgrounds and that it is effective with patients of varying ages and genders.

3.3 Data-Preprocessing

The images from the dataset are obtained and a total of 1065 images are randomly selected because there is a limitation of hardware capability. The images will be resized to 250 x 250 px to increase the rate of the process and for the images file with different resolution to obtain a uniform size for convenience in analysing (Tian et al, 2021) and is considered as an accepted and the file extension of images of PNG will be converted into JPEG format.

3.4 Transfer Learning

Transfer learning is a process where knowledge and information obtained from the data is transferred from one area to another. It is the adaptation of features that is used to learn on the initial stage.

Convolutional neural networks are very useful in transfer learning. This is because the image is able to undergo transfer learning using this architecture to perform both feature extraction and feature selection as the architecture is able to process both these feature engineering. (Lecun et al., 2015).

As mentioned in the literature review, convolutional neural networks contain convolution layer, pooling layer. For the convolution layer, it is the process where the kernel is understood as a matrix of numbers, the inputted image will be transferred to the image to be transformed according to the value of the kernel and to be the feature map (Skalski, 2019). The formula (Skalski, 2019) of the convolution is calculated below:

$$G[m,n] = (f\ \times h)[m,n] = \sum_{j}\sum_{k} h[j,kf[m-j,n-k] \qquad (3.1)$$

f: $inputted\ image$

h: $kernel$

m: $row\ of\ matrix$

n: $column\ of\ matrix$

Figure 3-2 is an example of a process in the convolution layer.

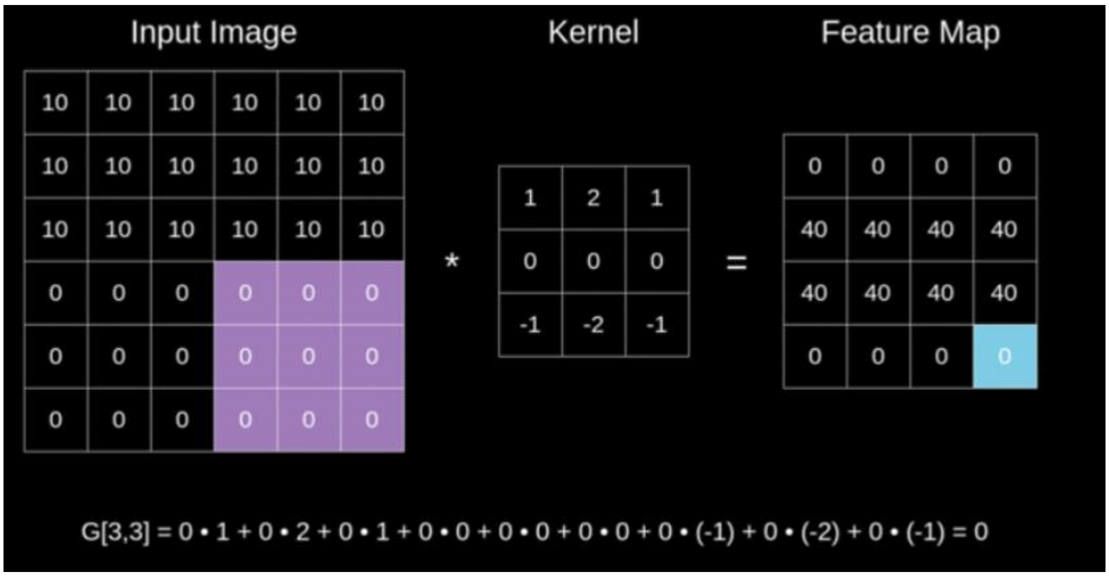


Figure 3-2 Process in the Convolution layer

Pooling layer is also used in convolutional neural networks. Images with feature maps extracted under this layer will get the size reduced while depth preserved to increase the speed used for the calculation (Dao, 2020). It is done by dividing the images into different regions and perform operations on the parts (Skalski, 2019). The formula (Khosla, 2021) of pooling layer is given below:

$$\frac{n_h - f + 1}{s} \times \frac{n_w - f + 1}{s} \times n_c \tag{3.2}$$

n_h: $height\ of\ feature\ map$
n_w: $width\ of\ feature\ map$
n_c: $number\ of\ channels\ in\ feature\ map$
f: $size\ of\ filter$
s: $length\ of\ stride$

Figure 3-3 is an example of a process in the pooling layer.

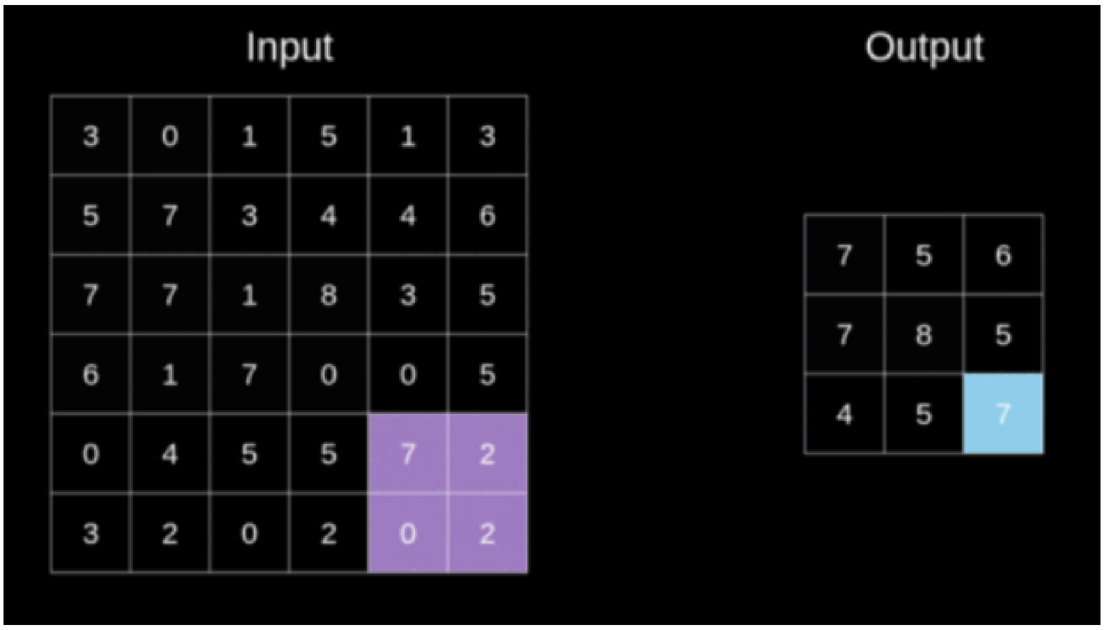


Figure 3-3 Process in the Pooling Layer

The type of the dataset used in this research is in image format, thus machine learning on image processing is used for the feature extraction of the datasets. For feature extraction and feature selection, the images will undergo four different methods for obtaining the result. The methods are VGG-11 (Simonyan & Kisserman, 2014), GoogLeNet (Szegedy et al, 2015), SqueezeNet v1.1 (Iandola et al, 2016) and Wide ResNet-50-2 (Zagoruyko & Komodakis, 2016).

Through literature review, we found out that VGG-11 contains many layers, which brings us the knowledge that the speciality of this network is that it is deep, and this brings significance to get an accurate result while training the model. VGG-11 is able to extract deep features due to having linear progressive convolutional layers together with small filter size.

Besides that, GoogLeNet which has various inception modules, for the increase in the number of parameters because of the abundance of filter size and its parallel convolutions. Thus, GoogLeNet is effective in extracting the features of the images available through the help of its inception module.

Additionally, SqueezeNet v1.1 has the advantage of being efficient during computation as the parameters are lesser for the number of layers it possesses. SqueezeNet v1.1 has the ability for deep feature extraction while the number of parameters is controlled with its ability of using squeeze and expand block.

Last but not least, WideResNet-50-2 models also have the advantage of being efficient during computation due to the parameters being lesser for the number of layers. WideResNet-50-2 is able to reduce vanishing gradient problems. This model's feature extraction is through the integration of the shallow features together with the deep features.

Therefore, these four pretrained models were selected in the current study to combine the different properties of each model and control the complexity of the computer. These previously tested models are used for the current binary classification problem and maintain estimates of the predicted probabilities of the images.

Table 3-1 Layer and Parameter of Proposed CNN Models

CNN	**Layers**	**Parameter**
VGG-11	11	30.15M
GoogLeNet	22	11.98M
SqueezeNet v1.1	14	0.72M
Wide ResNet-50-2	50	66.8M

These predicted scores will then undergo the Sugeno ensemble for identifying the final prediction score before entering and getting compared in the performance measure.

3.5 Sugeno Ensemble

Ensemble is an approach of merging two or more architectural designs. This ensemble method is able to produce a higher result due to each architectural design use having their own advantages and disadvantages, and thus ensemble will reduce the variance in the error during the prediction (Kundu et al., 2021).

In this study, instead of using predefined weight for every architecture, the fuzzy integral-based approach will consider the confidence for each of the architecture spontaneously without the constriction of needing to pre-define the weight, this is because this ensemble method can allocate a weight for those architectures while training.

Fuzzy integrals are effective in pattern recognition as they are able for problems related to aggregation which requires the use of fuzzy for measurement (Huang et al., 2021). These fuzzy measures will be important for the computation of the data aggregation for lung cancer.

For the calculation part, firstly we will need to assume that the Sugeno fuzzy-λ has a set of score $S = \{s_1, s_2, s_3, \ldots, s_n\}$ and we provide N as our number of architectures used, which is 4 in this case and $e \in S$. Now, proceed that the function of Sugeno fuzzy-λ measure which is $f_\lambda : 2^S \rightarrow [0, 1]$ which need to satisfy the condition stated below (Kundu et al., 2021).

1. $F_\lambda(S) = 1$
2. If $e_i \cap e_j = \varphi$, then $\exists$ a $\lambda > -1$, which will also show that the equation below is valid.

$$f_\lambda (e_i \cup e_j) = f_\lambda(e_i) + f_\lambda(e_j) + \lambda . f_\lambda(e_i)(e_j). \tag{3.3}$$

Through this, we can also determine the real root of λ is as given in Equation 3.4

$$\lambda + 1 = \prod_{n=1}^{N} (\lambda . \mathrm{f}(e_n + 1) \tag{3.4}$$

This fuzzy measure integral is defined as follows as a measurable space, and the measurable function with fuzzy measure is given in Equation 3.5.

$$\int f(x) d\Omega = \max\big(\min\big(f(x_i), \Omega(A_i)\big)\big) \tag{3.5}$$

3.6 Performance Measure and Comparison

The performance measure method that will be used for testing and evaluating the performance of this method, which is through using the accuracy metrics.

3.6.1 Accuracy Metrics

In this phase, the efficient of the proposed methods is determined and evaluated. This is because it is needed to detect the efficiency of this proposed method in the detection of lung cancer.

These results will be evaluated based on the confusion matrix and the result will be categorized into True Positive, False Positive, False Negative and True Negative. The confusion matrix can be better understood with Table 3-2.

Table 3-2 Confusion Matrix

		Actual Class	
		Positive (P)	Negative (N)
Predicted	Positive (P)	True Positive (TP)	False Positive (FP)
Class	Negative (N)	False Negative (FN)	True Negative (TN)

The difference between True Positive, False Positive, False Negative and True Negative are as follows:

- True Positive means that the method successfully predicted the correct result as positive
- False Positive means that the method falsely provides an incorrect result as positive.
- False Negative means that the method falsely provides an incorrect result as negative.
- True Negative means that the method successfully predicted the correct result as negative.

The measurement metrics, which are the precision, recall, F-Measure, and accuracy is calculated to determine the performance of the proposed method (Powers, 2011).

The definition of each measurement metrics are as follows:

- The accuracy is defined as the total number of correct predictions towards all instances.

$$Accuracy = \frac{Correct\ Detection\ (TP + TN)}{All\ Detection\ (TP + TN + FP + FN)} \quad (3.6)$$

- The precision, also called positive predictive value is determined by the number of correct predictions that are successfully detected which is then compared with all predictions that are true classified instances. This can be illustrated by using lung cancer detection. Each of the precision values of 1 indicates that every patient that is successfully detected to possess lung cancer truly has lung cancer.

$$Precision = \frac{True\ Positive(TP)}{Total\ Positive\ Prediction(TP + FP)} \quad (3.7)$$

- The recall, also called sensitivity is also determined by the number of correct predictions compared to all classifiers that are actually true. This can be illustrated by using lung cancer detection. Every recall value of 1 indicates that every patient that is successfully diagnosed with lung cancer will be compared with patients that are not diagnosed with lung cancer but actually have lung cancer.

$$Recall = \frac{True\ Positive\ (TP)}{Total\ Actual\ Positive\ (TP + FN)} \tag{3.8}$$

- The F-Measure aims to achieve a harmonic mean between recall and precision through the combination of both statements

$$F\ Measure = 2 \times \frac{Precision\ \times Recall}{Precision + Recall} \tag{3.9}$$

CHAPTER 4

RESEARCH IMPLEMENTATION

4.1 Introduction

Based on the methodology that was proposed in Chapter 3, this chapter presents the findings on relevant results and discussion on Lung Cancer Detection on CT Scan Images using Deep Learning Methods: Sugeno Fuzzy Integral Based CNN Ensemble. The very last thing that needs to be done is assessing the data in order to evaluate the correctness of the findings obtained using the Sugeno ensemble approach in combination with four CNN models (VGG-11, Wide ResNet-50-2, GoogLeNet, or SqueezeNet v1.1).

The accuracy metrics, which included accuracy, precision, recall, and F1-score that were used in the analysis of the model's performance to determine its usefulness. In this section, the outcomes of the model will be presented in order to determine whether or not Sugeno ensemble methods are helpful with CNN models for the identification of lung cancer.

4.2 Data Sample

As indicated in Chapter 3.2, the IQ-OTH/NCCD lung cancer dataset is a type of secondary data gathered from Iraq-Oncology Teaching Hospital/National Center for Cancer Diseases (Kareem, 2020).

Using an extension of the JPG format, 1065 images are selected at random from the total dataset and stored in the utilised dataset. These images are then separated into benign, malignant, and normal categories.

This dataset is the one that has been specifically picked and selected owing to the fact that the dataset has already been pre-categorized and there is no more work that needs to be done manually. In addition to this, the image format can be accessed immediately, and there is also no requirement for the file to be converted. Therefore, these two primary considerations play a significant role in the decision to adopt this dataset because it assists in saving a significant amount of time and work when it comes to classifying the images.

The dataset is then programmed for deep learning in this work, and the images are partitioned into their respective categories. This is due to the fact that it is important for them to be taught and validated in order to assess the effectiveness of the proposed method.

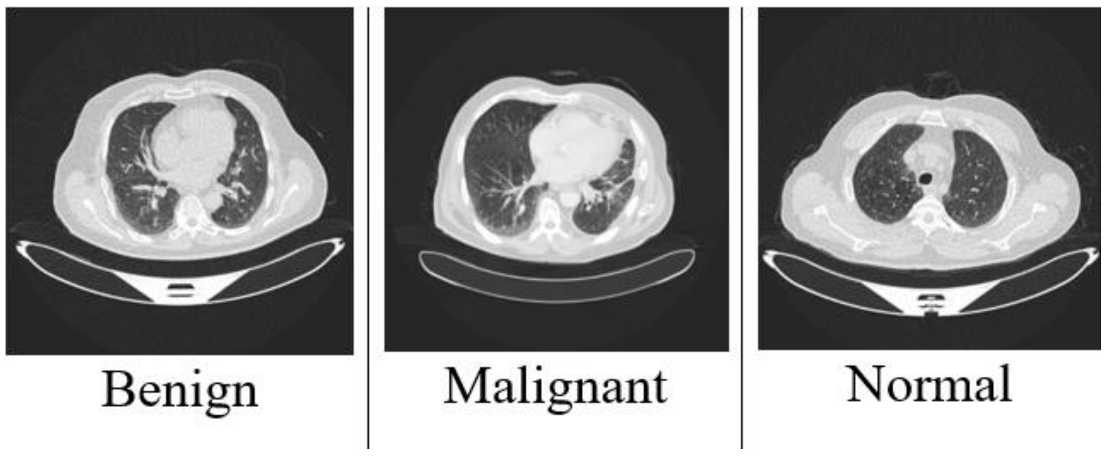


Figure 4-1 Type of Lung Cancer

4.3 Experimental Setup

The user interface is built and developed in such a way that takes into account the necessary steps for the suggested approach so that the study can be visualised for an easier understanding.

The system's homepage explains all the necessary procedures for deep learning to begin, including Split Data and Deep Learning. Cancer Detection is utilized to identify individual CT-scan images of the lungs to test for the result.

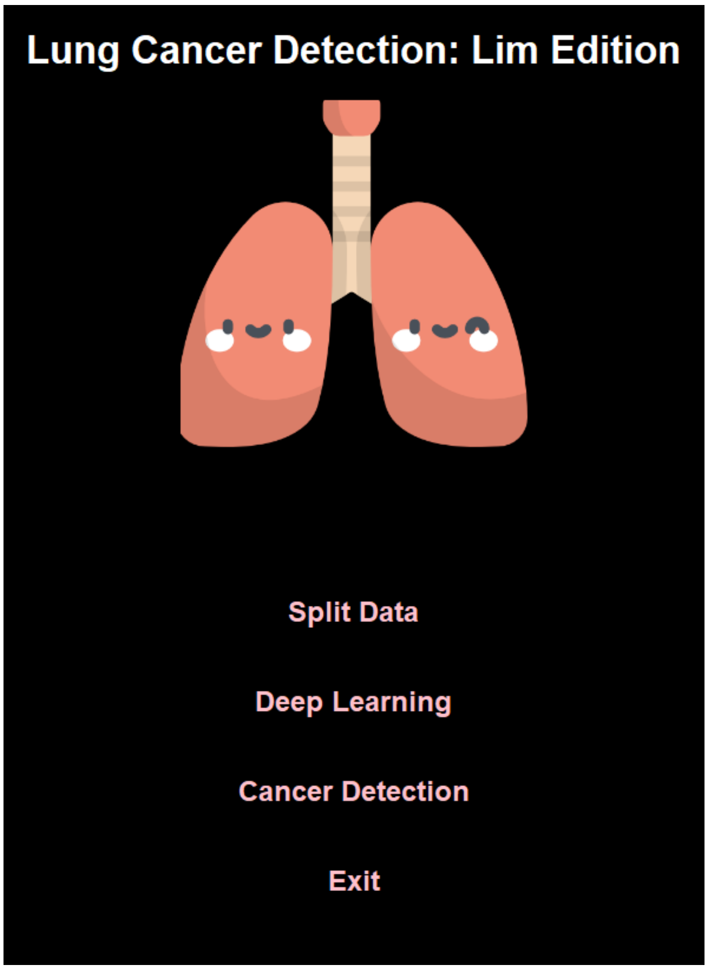


Figure 4-2 Homepage of Proposed Approach

Following the user's selection of the Split Data Button, the page will advance to this page, where they will discover the option to Choose Folder.

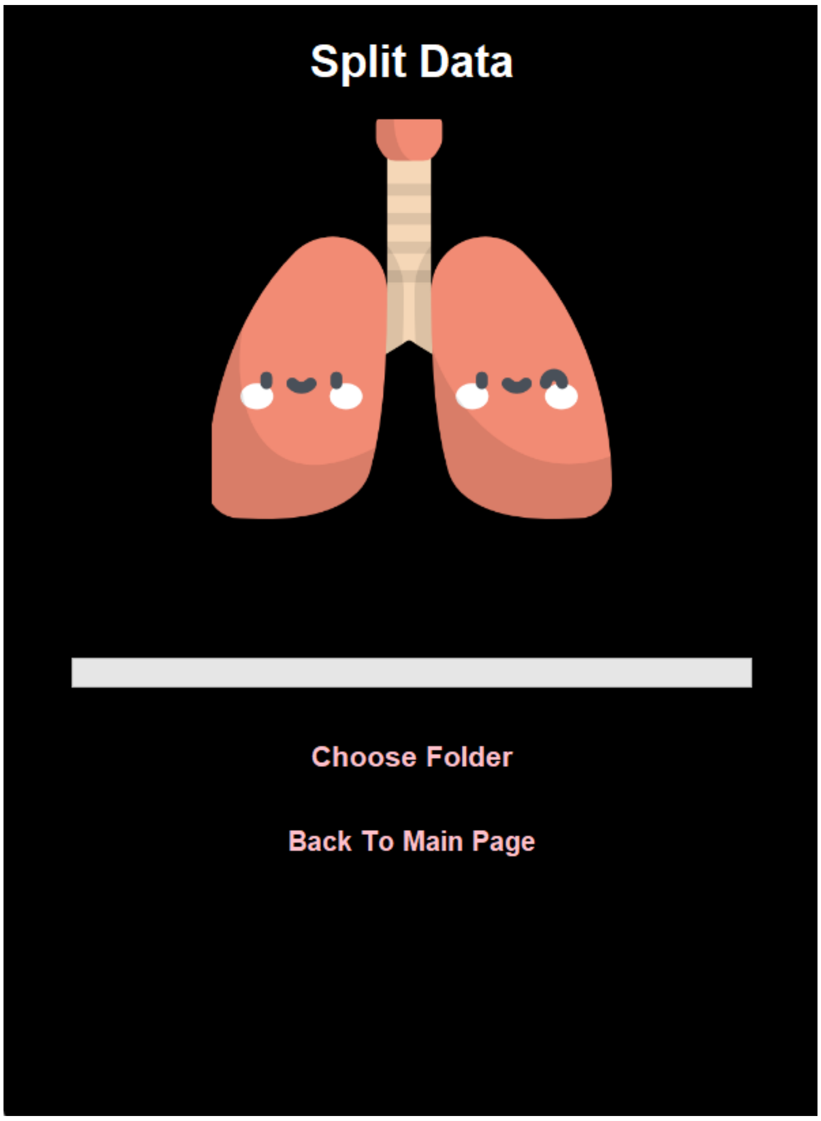
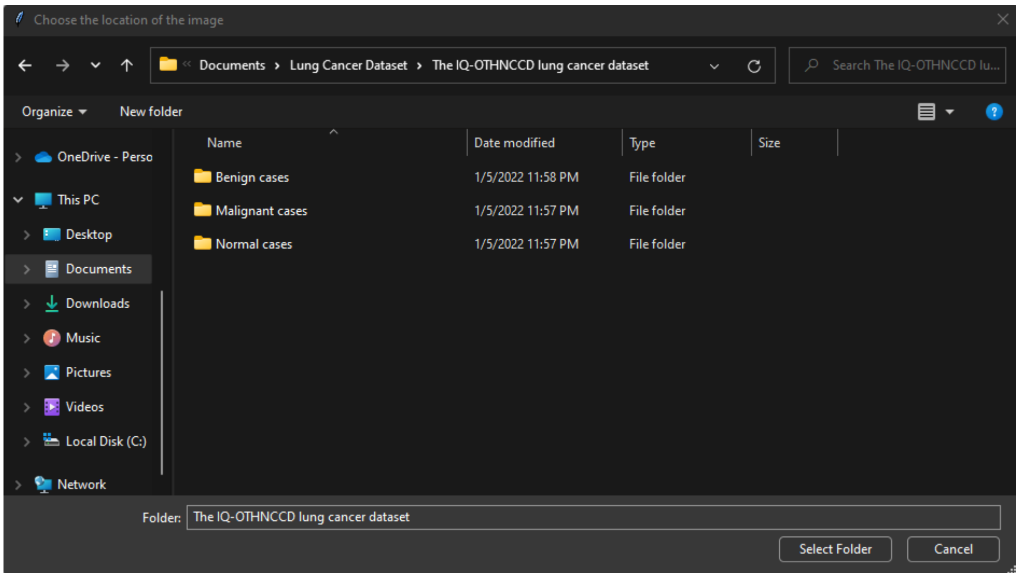


Figure 4-3 Split Data Page of Proposed Approach

After pressing the Choose Folder button, a prompt will appear for selecting the folder of where the images are stored so that the data can be split. After reaching the directory of the dataset that is needed for data pre-processing, in which to be split into the ratio of 80:20, the button of Select Folder is pressed.


Choose the location of the image
Documents › Lung Cancer Dataset › The IQ-OTHNCCD lung cancer dataset
Search The IQ-OTHNCCD lu...
Organize
New folder
OneDrive - Perso
This PC
Desktop
Documents
Downloads
Music
Pictures
Videos
Local Disk (C:)
Network
Select Folder
Cancel

Name	Date modified	Type	Size
Benign cases	1/5/2022 11:58 PM	File folder	
Malignant cases	1/5/2022 11:57 PM	File folder	
Normal cases	1/5/2022 11:57 PM	File folder	

Folder: The IQ-OTHNCCD lung cancer dataset

Figure 4-4 Choose Folder Interface of Proposed Approach

The progress bar will start moving, and once it reaches the finishing line, this means that the process of data splitting is halted and executed successfully.

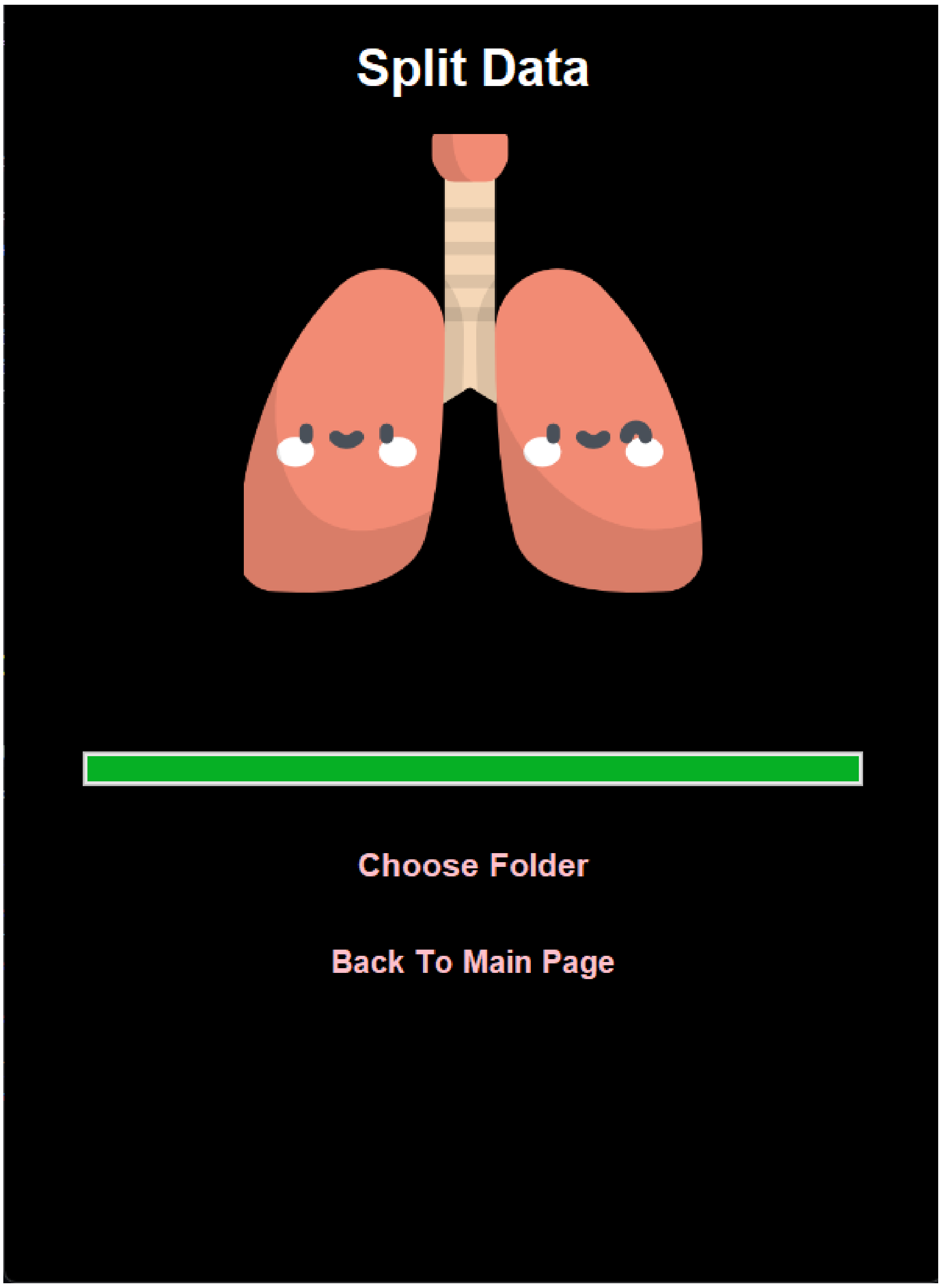


Figure 4-5 Split Page of Proposed Approach

After the completion of Split Data, the next step will be Deep Learning. This page is where the training of models and ensembles are performed. The steps and the final result will be showed on this page.

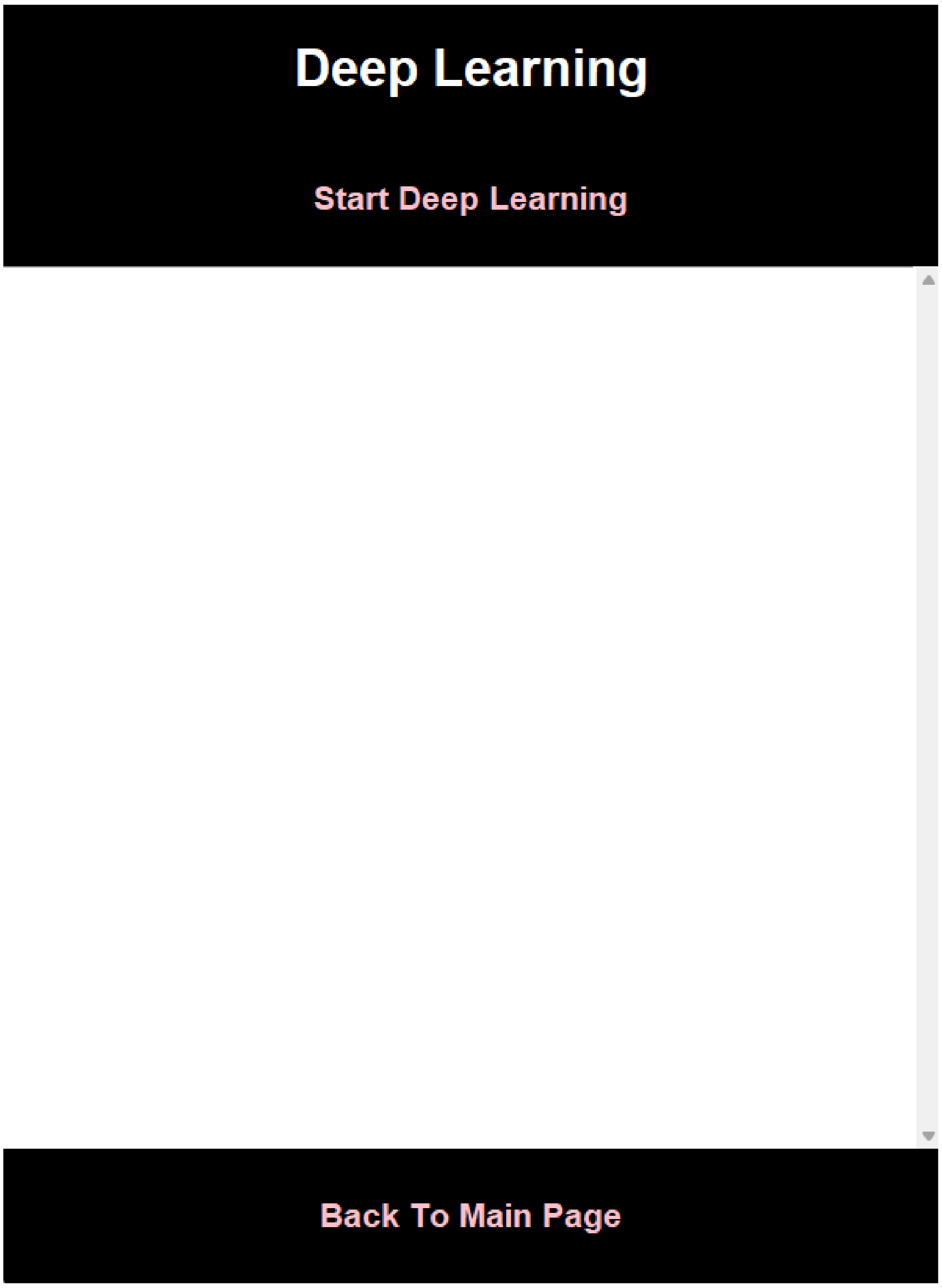
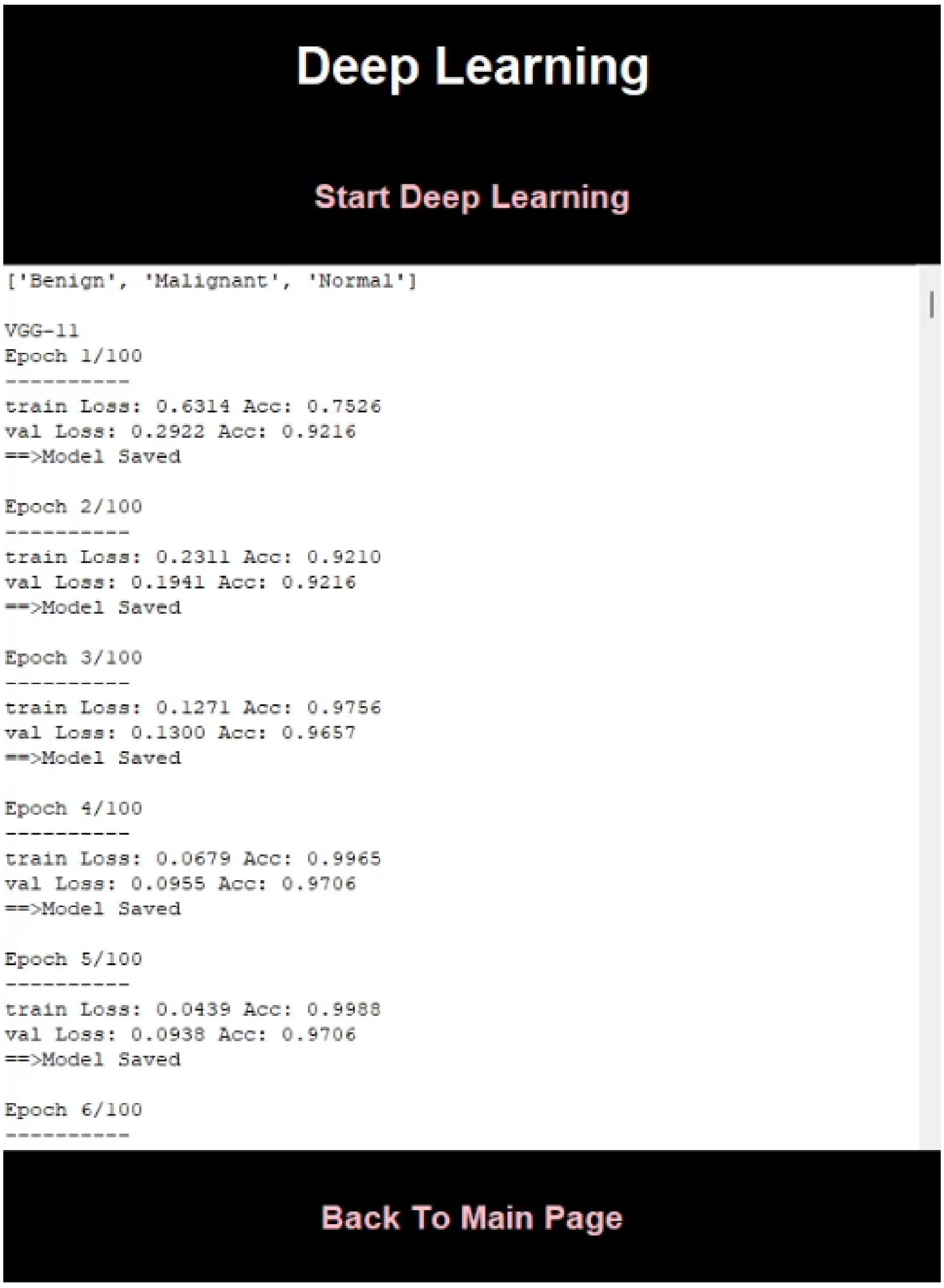


Figure 4-6 Deep Learning Page of Proposed Approach

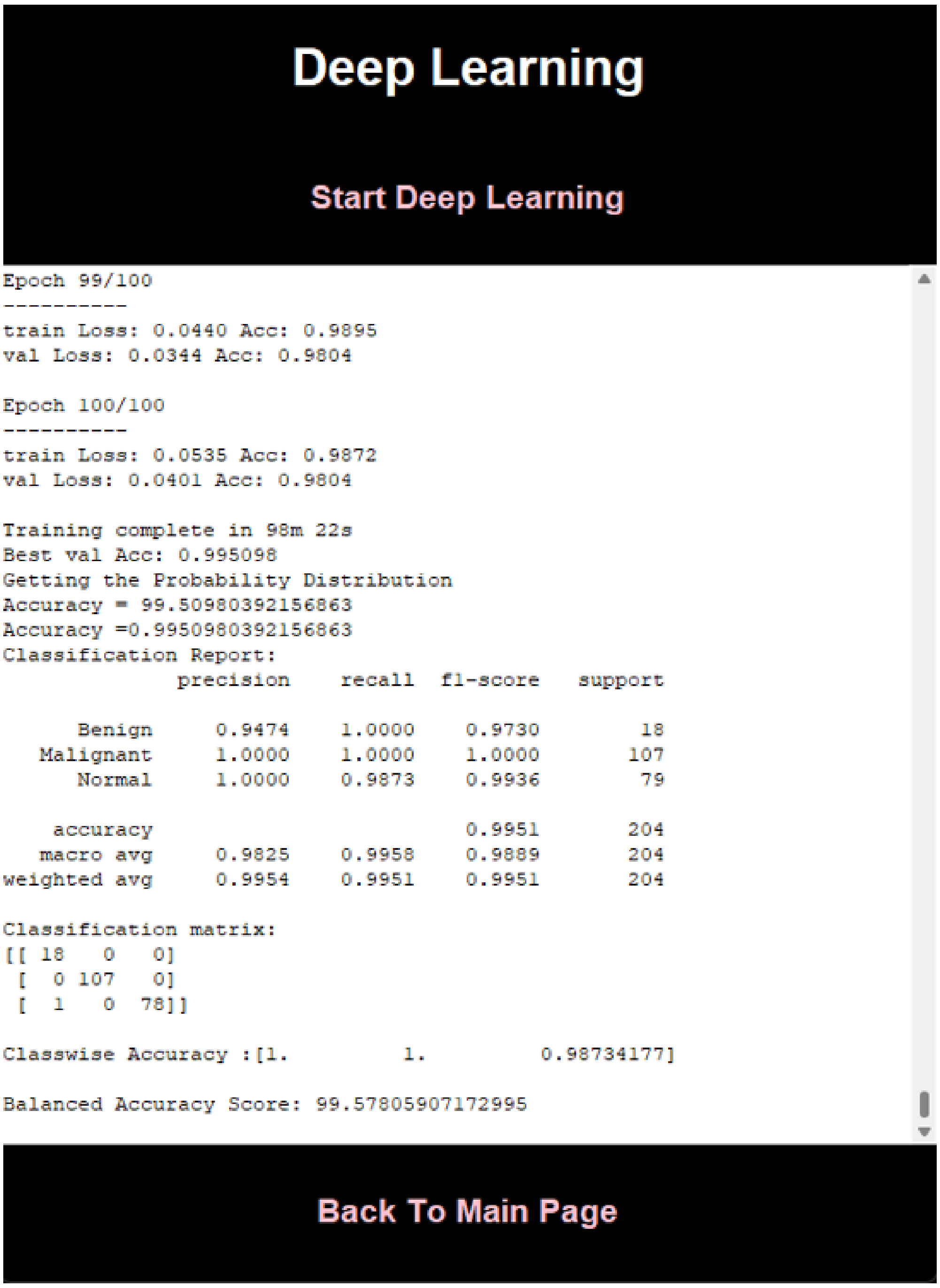
The Start Deep Learning button is then pressed to start the process of training the models together with the ensemble. Each of the epochs will start running and provide a result of accuracy. Throughout each epoch that has been performed, a better epoch result will overwrite the previous best result until the final best result is obtained.



```
['Benign', 'Malignant', 'Normal']

VGG-11
Epoch 1/100
----------
train Loss: 0.6314 Acc: 0.7526
val Loss: 0.2922 Acc: 0.9216
==>Model Saved

Epoch 2/100
----------
train Loss: 0.2311 Acc: 0.9210
val Loss: 0.1941 Acc: 0.9216
==>Model Saved

Epoch 3/100
----------
train Loss: 0.1271 Acc: 0.9756
val Loss: 0.1300 Acc: 0.9657
==>Model Saved

Epoch 4/100
----------
train Loss: 0.0679 Acc: 0.9965
val Loss: 0.0955 Acc: 0.9706
==>Model Saved

Epoch 5/100
----------
train Loss: 0.0439 Acc: 0.9988
val Loss: 0.0938 Acc: 0.9706
==>Model Saved

Epoch 6/100
----------
```

Figure 4-7 Epoch Training in the Proposed Approach

The final result of the proposed approach is then obtained using the Sugeno Fuzzy Integral upon completion.



```
Epoch 99/100
----------
train Loss: 0.0440 Acc: 0.9895
val Loss: 0.0344 Acc: 0.9804

Epoch 100/100
----------
train Loss: 0.0535 Acc: 0.9872
val Loss: 0.0401 Acc: 0.9804

Training complete in 98m 22s
Best val Acc: 0.995098
Getting the Probability Distribution
Accuracy = 99.50980392156863
Accuracy =0.9950980392156863
Classification Report:
              precision    recall  f1-score   support

      Benign     0.9474    1.0000    0.9730        18
   Malignant     1.0000    1.0000    1.0000       107
      Normal     1.0000    0.9873    0.9936        79

    accuracy                         0.9951       204
   macro avg     0.9825    0.9958    0.9889       204
weighted avg     0.9954    0.9951    0.9951       204

Classification matrix:
[[ 18   0   0]
 [  0 107   0]
 [  1   0  78]]

Classwise Accuracy :[1.         1.         0.98734177]

Balanced Accuracy Score: 99.57805907172995
```

Figure 4-8 Final Result of the Proposed Approach

Last but not least, there is a bonus feature of the system which allows one to determine the result of a single image with the help of the pretrained model, which is the button of Cancer Detection.

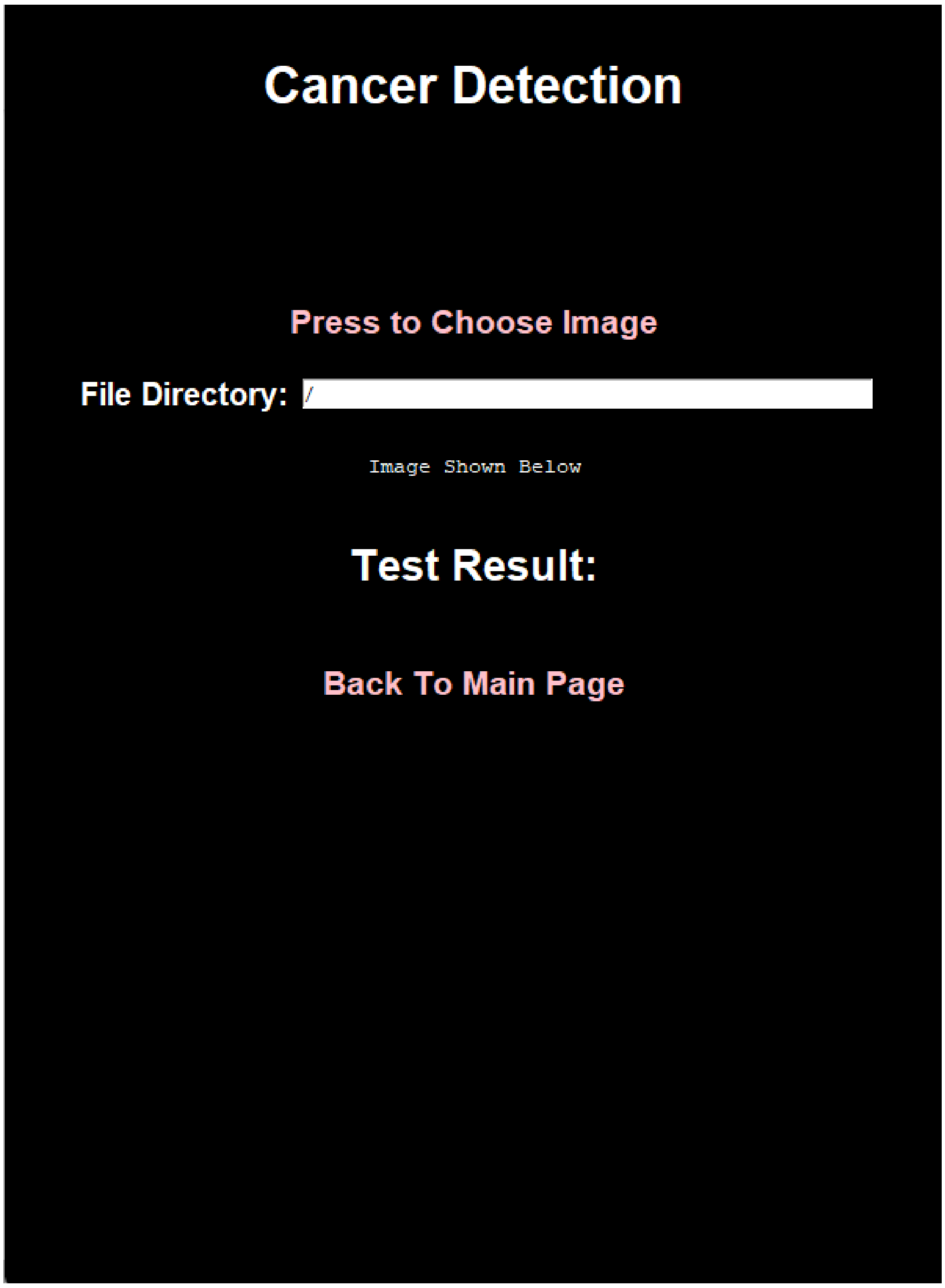
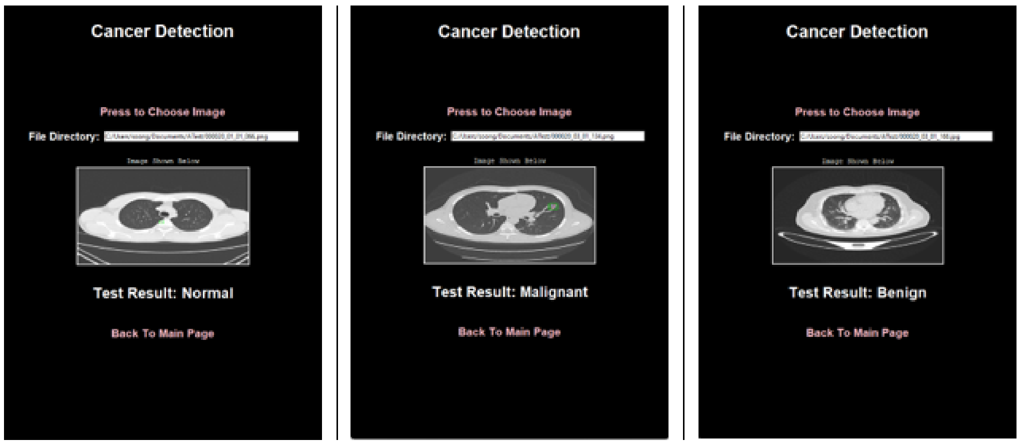


Figure 4-9 Cancer Detection of Proposed Approach

The result will be shown clearly to show the answer of the final detection made by the pretrained models. From the figure, there are three types of different results that will be shown, which are Normal, Malignant and Benign.

This concludes the proposed approach which is turned into a fully functional user interface for easier access and navigation through the steps that are needed for making a fully-fledged system



Test Result: Normal

Test Result: Malignant

Test Result: Benign

Figure 4-10 Result obtained from Cancer Detection

4.4 Discussion

The study was conceived with the goal of including all of the prerequisite activities for deep learning in image processing. These procedures entail separating the data and training using the dataset.

Firstly, the IQ-OTH/NCCD lung cancer dataset (Kareem, 2020) is selected and downloaded from the website Kaggle.com. 1065 images were extracted randomly from the entire dataset. The dataset will first undergo data splitting, which follows the standard test-train ratio of 80:20 as it is the ideal ratio for training and validating the dataset. The 80% of the dataset will be distributed for training purposes and the remaining 20% will be distributed for validation purposes.

The dataset will be trained with GoogLeNet, SqueezeNet v1.1, Wide ResNet-50-2, and VGG-11 after being separated into training and validation. These models each have their own good traits, which can be used to compensate for the others' flaws. GoogleNet's Inception modules can detect features at several scales using convolutions with different filter sizes. SqueezeNet v1.1 has fire modules with a squeeze layer and an expansion layer to reduce file size while maintaining precision. Wide ResNet-50-2 has a residual block that can avoid the vanishing gradient problem. VGG-11 has many weight layers to boost performance.

After each of the four models has successfully finished their training, the trained model will undergo an ensemble, which is a Sugeno fuzzy integral. Sugeno fuzzy integral examines each model's degree of confidence before allocating a weight to any of the models.

Last but not least, the accuracy of the models are then calculated using the confusion matrix to measure the performance of the proposed approach compared to existing models.

CHAPTER 5

RESULT AND DISCUSSION

5.1 Evaluation of Results

. The final result is obtained after the deep learning process using the proposed approach. There are various results that are obtained which includes the accuracy, precision, recall and F1-score.

In start, the four models that are used are VGG-11, GoogLeNet, SqueezeNet v1.1 and Wide ResNet-50-2. The accuracy of the models are as follows.

Table 5-1 Accuracy and Time Taken of CNN Models

Model	**Accuracy (%)**	**Time Taken**
VGG-11	97.54	62m 29s
GoogLeNet	99.02	32m 52s
SqueezeNet v1.1	91.67	32m 25s
Wide ResNet-50-2	99.51	99m 22s

According to the data presented in the table that is located above, the levels of accuracy that were achieved by VGG-11, GoogLeNet, SqueezeNet v1.1, and Wide ResNet-50-2 are respectively 97.54 %, 99.02 %, 91.67 %, and 99.51 %. Comparatively, the times it takes to complete the identical models are 62 minutes and 29 seconds, 32 minutes and 52 seconds, 32 minutes and 25 seconds, and 99 minutes and 22 seconds.

The data that were gathered allowed us to determine that the model with the highest accuracy rate is Wide ResNet-50-2. This model was able to achieve an accuracy rate of 99.51 %, making it the most accurate model. The next most accurate resource is GoogleNet, at 99.02 %, followed by here. The VGG-11 came in second with an accuracy of 97.54 % after that. The model known as SqueezeNet v1.1 has an accuracy of 91.67 %, making it the model with the lowest accuracy.

It took SqueezeNet v1.1 32 minutes and 25 seconds to complete, making it the model with the quickest time. GoogLeNet came in second with 32 minutes and 52 seconds, and VGG-11 came in third with 62 minutes and 29 seconds. The model that took the longest time was Wide ResNet-50-2, which took a total of 99 minutes and 22 seconds to complete.

Following the execution of the Sugeno fuzzy ensemble procedure on the four CNN models, the confusion matrix for the result along with the final result is acquired.

		Actual Class		
		Benign	Malignant	Normal
Predicted Class	Benign	18	0	0
	Malignant	0	107	0
	Normal	1	0	78

Figure 5-1 Result of Confusion Matrix of Proposed Approach

Table 5-2 Result of Proposed Approach

Confusion Metrics	Score (%)
Accuracy	99.58
Precision	99.54
Recall	99.51
F1-Score	99.58

The proposed approach achieved a score of 99.58% in terms of accuracy, 99.54% in terms of precision, 99.51% in terms of recall, and 99.58% in terms of F1-score.

5.2 Discussion

. Before the start of the discussion, the layers of each of the CNN models are recorded and stated to help in discussion purposes.

Table 5-3 Number of Layer of CNN Models

Models	Number Of Layer
VGG-11	11
GoogLeNet	22
SqueezeNet v1.1	14
Wide ResNet-50-2	50

The accuracy of each and every CNN model that has been used will be investigated and analysed in this study. To begin, the number of layers plays a significant part in the growth in accuracy as Wide ResNet-50-2 has the maximum number of layers, which is 50 layers, and it received a result of 99.58%. GoogLeNet came in second position with 22 layers and an accuracy of 99.02%. The one and only exception to this rule is that the VGG-11 model, which has 11 layers, obtained an accuracy of 97.54% and it managed to outperform the SqueezeNet v1.1 model with an accuracy of 91.67%, which has 14 layers.

After finishing training and validating all the models, the Sugeno fuzzy ensemble is executed on the four CNN models and the results are then calculated and tabulated.

Sugeno fuzzy ensemble has provided the final result using its characteristic of providing weightage on the fly. The result of the final accuracy, precision, recall and F1-score is then recorded.

CHAPTER 6

CONCLUSIONS

6.1 Introduction

This section mentioned about the contribution of the study, the fulfilment of the objective of the study, the constraint of the study and future suggestions according this studies.

6.2 Research Contribution

The purpose of this study is to provide a new approach for the identification of lung cancer by making use of the Sugeno Fuzzy Integral. This approach should be able to differentiate between benign, malignant, and normal lung CT-images while also achieving a high level of accuracy.

6.2.1 Fulfillment of the Research Objective

This study makes a contribution that is threefold by achieving the purpose that was expressed in accordance with its relevance in Section 1.4. The successful completion of each objective will be brought to light in the proper manner so that the accomplishment of each goal may be highlighted. The results of this study can help medical professionals diagnose and categorise patients who are suffering from lung cancer. This is a useful application of the findings. The findings have important repercussions for medical professionals and the medical industry as a whole in the field of artificial intelligence.

Objective 1: To study the accuracy of Sugeno fuzzy ensemble combined with CNN models to assist in lung cancer detection.

The project began with the investigation of the proposed approach for the detection of lung cancer. The first goal was accomplished by writing code for the Sugeno Fuzzy Integral Ensemble algorithm. This allowed for the goal to be considered complete. The approaches are introduced in the methodology and gaining accuracy by utilising the lung cancer dataset from IQ/OTHNCCD are described, as is explained in Chapter 3. The procedure had an accuracy of 99.58% when applied to the classification of identifying benign, malignant as well as normal lungs from CT scan images.

This demonstrates that the first objective was accomplished in a successful manner.

Objective 2: To evaluate whether Sugeno fuzzy ensemble combined with CNN models outperform existing methods.

Following the successful completion of the first objective, the second objective is carried out, which is to assess whether or not this approach outperforms other existing studies that are already being used. The comparison with the existing studies as compared below.

Table 6-1 Comparison with Existing Studies

Comparison with Existing Studies	
Title	**Accuracy (%)**
Proposed Approach	**99.58**
Comparison of deep learning convolutional neural network (CNN) architectures for CT lung cancer classification	GoogLeNet: 94.53 SqueezeNet: 94.13 ShuffleNet: 92.91 DenseNet: 93.52 MobileNetV2: 92.91
Diagnosis of lung cancer based on CT scans using CNN	93.55
Lung Nodule Detection in Computed Tomography Scans Using Deep Learning	96.00
Transfer learning with GoogLeNet for detection of lung cancer	94.38

From the best of my knowledge, this study managed to outperform all existing studies that are currently available. Some of the existing studies are shown in the table with their accuracy. This study to prove that it has successfully outperformed existing methods that are available currently.

This demonstrates that the second objective was accomplished in a successful manner.

Objective 3: To develop a prototype of lung cancer detection using Sugeno fuzzy ensemble combined with CNN models.

This objective was created specifically to make the study more accessible to those that are not versed in viewing Python codes. As the algorithm was developed using Python, the graphical user interface was designed using the Tkinter library for easier use, which can be shown in Section 4.3.

This demonstrates that the third objective was accomplished in a successful manner.

6.3 System Constraint

Despite the substantial contributions made by the study, the methodology and strategy employed in the study have a number of inherent limitations. Consequently, additional study and investigation are necessary to remedy these deficiencies.

To begin, the entire amount of time spent on the training is 3 hours, 47 minutes, and 6 seconds, which contributes to the length of time needed for the study to be completed. These are the result of the fact that 4 models are required for training, and each model has to be trained separately prior to the models going through the ensemble process using Sugeno fuzzy. Each model requires an average of around 40 minutes and this contributes to a long duration of deep learning.

Secondly, the study is also unable to differentiate the stages of lung cancer. The main reasons that contribute to this constriction is due to the lack of dataset for training purposes. The study is only able to differentiate the type of lung cancer but are unable to detect the stages of lung cancer if lung cancer is present.

Last but not least, this study does not take in considering other epochs for training. The only epoch available is 100 epochs, which is commonly used for deep learning.

6.4 Future Suggestions

There are a lot of different aspects that can be improved upon for future studies in order to get better results from deep learning.

To begin with, one of the suggestions that needs to be made is to shorten the amount of time spent training. According to Section 6.3, the fact that this study makes use of four models contributes to the length of time required for training and validation. GoogLeNet and Wide ResNet-50-2 are the two models that have the potential to be used in subsequent research since they have the highest level of accuracy. These two models may be trained and put through an ensemble via the use of Sugeno Fuzzy. These may generate results that are comparable, but they cut the amount of time needed for training in half.

Second, the use of datasets that contain information on various stages of lung cancer in upcoming research can also be of use in the early detection of lung cancer. Patients may get new insights from this study. Medical professionals may be better able to treat them as this allows them to better comprehend the stages of the disease from which they are now suffering as a result.

Last but not least, different epochs such as an increase of an epoch to 500 or 1000 epochs or decrease in epochs such as 50 epochs may be utilized to determine whether the difference in epoch will affect the accuracy of the proposed algorithm.

6.5 Summary

In conclusion, the objectives of this project have been fulfilled since all the objectives have been successfully achieved. However, there are still limitations from these studies that may bring inspiration for future studies to be conducted in order to overcome these shortcomings in the upcoming future.

REFERENCES

Al-Huseiny, M. S., & Sajit, A. S. (2021). Transfer learning with googlenet for detection of lung cancer. *Indonesian Journal of Electrical Engineering and Computer Science*, *22*(2), 1078. https://doi.org/10.11591/ijeecs.v22.i2.pp1078-1086

Al-Yasriy, H. F., AL-Husieny, M. S., Mohsen, F. Y., Khalil, E. A., & Hassan, Z. S. (2020). Diagnosis of lung cancer based on CT scans using CNN. *IOP Conference Series: Materials Science and Engineering*, *928*(2), 022035. https://doi.org/10.1088/1757-899x/928/2/022035

Alhayani, B., Abbas, S. T., Mohammed, H. J., & Mahajan, H. B. (2021). Intelligent secured two-way image transmission using Corvus Corone module over WSN. *Wireless Personal Communications*, *120*(1), 665–700. https://doi.org/10.1007/s11277-021-08484-2

Almas, B., Sathesh, K., & Rajasekaran, S. (2019). A deep analysis of Google Net and AlexNet for lung cancer detection. *International Journal of Engineering and Advanced Technology*, *9*(2), 395–399. https://doi.org/10.35940/ijeat.b3226.129219

Armato, S. G., McLennan, G., Bidaut, L., McNitt-Gray, M. F., Meyer, C. R., Reeves, A. P., Zhao, B., Aberle, D. R., Henschke, C. I., Hoffman, E. A., Kazerooni, E. A., MacMahon, H., van Beek, E. J., Yankelevitz, D., Biancardi, A. M., Bland, P. H., Brown, M. S., Engelmann, R. M., Laderach, G. E., … Clarke, L. P. (2011). The Lung Image Database Consortium (LIDC) and Image Database Resource Initiative (IDRI): A completed reference database of lung nodules on CT scans. *Medical Physics*, *38*(2), 915–931. https://doi.org/10.1118/1.3528204

Arya, D., Maeda, H., Ghosh, S. K., Toshniwal, D., Mraz, A., Kashiyama, T., & Sekimoto, Y. (2021). Deep learning-based road damage detection and classification for multiple countries. *Automation in Construction*, *132*, 103935. https://doi.org/10.1016/j.autcon.2021.103935

Ashhar, S. M., Mokri, S. S., Rahni, A. A., Huddin, A. B., Zulkarnain, N., Azmi, N. A., & Mahaletchumy, T. (2021). Comparison of deep learning Convolutional Neural Network (CNN) architectures for CT Lung Cancer Classification. *International Journal of Advanced Technology and Engineering Exploration*, *8*(74), 126–134. https://doi.org/10.19101/ijatee.2020.s1762126

Banerjee, P. (2021, November 6). *Top 5 programming languages and their libraries for machine learning in 2020*. GeeksforGeeks. Retrieved November 9, 2021, from https://www.geeksforgeeks.org/top-5-programming-languages-and-their-libraries-for-machine-learning-in-2020/

Bertolini, R., Finch, S. J., & Nehm, R. H. (2021). Testing the impact of novel assessment sources and machine learning methods on predictive outcome modeling in Undergraduate Biology. *Journal of Science Education and Technology*, *30*(2), 193–209. https://doi.org/10.1007/s10956-020-09888-8

Boslaugh, S. (2007). *Secondary data sources for Public Health: A Practical Guide*. Cambridge University Press.

Buckner, C. (2019). Deep learning: A philosophical introduction. *Philosophy Compass*, *14*(10). https://doi.org/10.1111/phc3.12625

Cancer.Net. (2019, December 5). *What is metastasis?* Cancer.Net. Retrieved November 2, 2021, from https://www.cancer.net/navigating-cancer-care/cancer-basics/what-metastasis

Chen, C., Wu, W., Chen, C., Chen, F., Dong, X., Ma, M., Yan, Z., Lv, X., Ma, Y., & Zhu, M. (2021). Rapid diagnosis of lung cancer and glioma based on serum Raman spectroscopy combined with deep learning. *Journal of Raman Spectroscopy, 52*(11), 1798–1809. https://doi.org/10.1002/jrs.6224

Cheng, K., Khokhar, M. S., Ayoub, M., & Jamali, Z. (2020). Nonlinear dimensionality reduction in Robot Vision for industrial monitoring process via deep three dimensional Spearman Correlation Analysis (D3D-SCA). *Multimedia Tools and Applications*, *80*(4), 5997–6017. https://doi.org/10.1007/s11042-020-09859-6

Corno, A. F., & Festa, G. P. (2009). Introduction to CT scan and MRI. *Congenital Heart Defects*, 1–17. https://doi.org/10.1007/978-3-7985-1719-6_1

Cortes, C., & Vapnik, V. (1995). Support-Vector Networks. *Machine Learning, 20*(3), 273–297. https://doi.org/10.1007/bf00994018

da Silva, E. A. B., & Mendonça, G. V. (2005). Digital Image Processing. *The Electrical Engineering Handbook*, 891–910. https://doi.org/10.1016/b978-012170960-0/50064-5

Dao, H. (2020). *Image Classification Using Convolutional Neural Networks* (thesis).

Daugman, J. G. (1988). Complete discrete 2-d gabor transforms by Neural Networks for image analysis and compression. *IEEE Transactions on Acoustics, Speech, and Signal Processing, 36*(7), 1169–1179. https://doi.org/10.1109/29.1644

Dhyani, M., & Kumar, R. (2021). An intelligent chatbot using deep learning with bidirectional RNN and attention model. *Materials Today: Proceedings*, *34*, 817–824. https://doi.org/10.1016/j.matpr.2020.05.450

Dobko, M. (2019). Lung Nodule Detection in Computed Tomography Scans Using Deep Learning. https://doi.org/https://s3.eu-central-1.amazonaws.com/ucu.edu.ua/wp-content/uploads/sites/8/2019/12/Mariia-Dobko.pdf

El Naqa, I., & Murphy, M. J. (2015). What is machine learning? *Machine Learning in Radiation Oncology,* 3–11. https://doi.org/10.1007/978-3-319-18305-3_1

Fradkov, A. L. (2020). Early history of machine learning. *IFAC-PapersOnLine*, *53*(2), 1385–1390. https://doi.org/10.1016/j.ifacol.2020.12.1888

Gindi, A. M. A., Attiatalla, T. A., & Mostafa, M.-S. M. (2014). A Comparative Study for Comparing Two Feature Extraction Methods and Two Classifiers in Classification of Early-stage Lung Cancer Diagnosis of chest x-ray images. *Journal of American Science 10(6).*

Gonzalez, R. C., Eddins, S. L., & Woods, R. E. (2009). *Digital Image Processing using Matlab*. Pearson Education.

Guo, Y., Liu, Y., Oerlemans, A., Lao, S., Wu, S., & Lew, M. S. (2016). Deep Learning for Visual Understanding: A Review. *Neurocomputing*, *187*, 27–48. https://doi.org/10.1016/j.neucom.2015.09.116

He, K., Zhang, X., Ren, S., & Sun, J. (2016). Deep residual learning for image recognition. *2016 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*. https://doi.org/10.1109/cvpr.2016.90

Huang, J.-X., Huang, Y.-L., Hsieh, C.-Y., & Wei, C.-S. (2021). Fuzzy integral with particle swarm optimization for CNN-based motor-imagery EEG classification. https://doi.org/10.1101/2021.09.07.459275

Iandola , F. N., Han, S., Moskewicz, M. W., Ashraf, K., Dally, W. J., & Keutzer, K. (2016). SqueezeNet: AlexNet-level accuracy with 50x fewer parameters and <0.5MB model size. *ArXiv:1602.07360.*

IBM Cloud Education. (2020, July 15). *What is machine learning?* IBM. Retrieved December 1, 2021, from https://www.ibm.com/my-en/cloud/learn/machine-learning

Ifrah, G. (2001). The Universal History of Computing: From the abacus to the Quantum Computer. *Choice Reviews Online*, *38*(09). https://doi.org/10.5860/choice.38-5056

Izadkhah, H. (2022). Popular deep learning image classifiers. *Deep Learning in Bioinformatics*, 215–247. https://doi.org/10.1016/b978-0-12-823822-6.00016-0

Jung, J., Maeda, M., Chang, A., Bhandari, M., Ashapure, A., & Landivar-Bowles, J. (2021). The potential of remote sensing and artificial intelligence as tools to improve the resilience of Agriculture Production Systems. *Current Opinion in Biotechnology*, *70*, 15–22. https://doi.org/10.1016/j.copbio.2020.09.003

Kalisz, K., Rassouli, N., Dhanantwari, A., Jordan, D., & Rajiah, P. (2018). Noise characteristics of virtual monoenergetic images from a novel detector-based spectral CT Scanner. *European Journal of Radiology*, *98*, 118–125. https://doi.org/10.1016/j.ejrad.2017.11.005

Kareem, H. F. (2020, May 24). *The IQ-oth/NCCD lung cancer dataset*. Kaggle. Retrieved December 1, 2021, from https://www.kaggle.com/hamdallak/the-iqothnccd-lung-cancer-dataset

Karthikeyan, B. (2012). Analysis of image segmentation for radiographic images. *Indian Journal of Science and Technology*, *5*(11), 1–5. https://doi.org/10.17485/ijst/2012/v5i11.9

Kelleher, J. D. (2019). *Deep learning*. The MIT Press.

Khosla, S. (2021, July 29). *CNN: Introduction to pooling layer*. GeeksforGeeks. Retrieved January 1, 2022, from https://www.geeksforgeeks.org/cnn-introduction-to-pooling-layer/

Krizhevsky, A., Sutskever, I., & Hinton, G. E. (2012). ImageNet classification with deep convolutional neural networks. *Proceedings of the 25th International Conference on Neural Information Processing Systems*, *1*, 1097–1105.

Kundu, R., Singh, P. K., Mirjalili, S., & Sarkar, R. (2021). Covid-19 detection from lung CT-scans using a fuzzy integral-based CNN ensemble. *Computers in Biology and Medicine*, *138*, 104895. https://doi.org/10.1016/j.compbiomed.2021.104895

LeCun, Y., Bengio, Y., & Hinton, G. (2015). Deep learning. *Nature*, *521*(7553), 436–444. https://doi.org/10.1038/nature14539

Lecun, Y., Bottou, L., Bengio, Y., & Haffner, P. (1998). Gradient-based learning applied to document recognition. *Proceedings of the IEEE*, *86*(11), 2278–2324. https://doi.org/10.1109/5.726791

Li, P., Wang, S., Li, T., Lu, J., HuangFu, Y., & Wang, D. (2020). A Large-Scale CT and PET/CT Dataset for Lung Cancer Diagnosis [Dataset] (Lung-PET-CT-Dx). *The Cancer Imaging Archive*. https://doi.org/https://doi.org/10.7937/TCIA.2020.NNC2-0461

Litjens, G., Kooi, T., Bejnordi, B. E., Setio, A. A., Ciompi, F., Ghafoorian, M., van der Laak, J. A. W. M., van Ginneken, B., & Sánchez, C. I. (2017). A survey on Deep Learning in medical image analysis. *Medical Image Analysis*, *42*, 60–88. https://doi.org/10.1016/j.media.2017.07.005

Lynne Eldridge, M. D. (2020, July 15). *How chest X-rays can sometimes misdiagnose lung cancer*. Verywell Health. Retrieved November 10, 2021, from https://www.verywellhealth.com/chest-x-rays-for-lung-cancer-diagnosis-4107046#:~:text=Obscured%20Images,start%20to%20clog%20the%20airways

Makaju, S., Prasad, P. W. C., Alsadoon, A., Singh, A. K., & Elchouemi, A. (2018). Lung cancer detection using CT scan images. *Procedia Computer Science*, *125*, 107–114. https://doi.org/10.1016/j.procs.2017.12.016

Mining, E. (2020). *Machine Learning for Beginners: A Complete and Phased Beginner's Guide to Learning and Understanding Machine Learning and Artificial Intelligence*. Everooks Ltd.

Mueller, J. P., & Massaron, L. (2021). *Machine learning For Dummies*. John Wiley et Sons, Inc.

Naeini, E. Z., & Prindle, K. (2018). Machine learning and learning from machines. *The Leading Edge*, *37*(12), 886–893. https://doi.org/10.1190/tle37120886.1

Najafabadi, M. M., Villanustre, F., Khoshgoftaar, T. M., Seliya, N., Wald, R., & Muharemagic, E. (2015). Deep learning applications and challenges in Big Data Analytics. *Journal of Big Data*, *2*(1). https://doi.org/10.1186/s40537-014-0007-7

National Cancer Institute. (2021, May 5). *What is cancer?* National Cancer Institute. Retrieved November 1, 2021, from https://www.cancer.gov/about-cancer/understanding/what-is-cancer

Neftci, E. O., & Averbeck, B. B. (2019). Reinforcement learning in artificial and Biological Systems. *Nature Machine Intelligence*, *1*(3), 133–143. https://doi.org/10.1038/s42256-019-0025-4

Pandey, B., Kumar Pandey, D., Pratap Mishra, B., & Rhmann, W. (2021). A comprehensive survey of deep learning in the field of medical imaging and Medical Natural Language Processing: Challenges and Research Directions. *Journal of King Saud University - Computer and Information Sciences*. https://doi.org/10.1016/j.jksuci.2021.01.007

Patterson, J., & Gibson, A. (2017). *Deep learning: A practitioner's approach*. O'Reilly.

Petrik, V., Apok, V., Britton, J. A., Bell, B. A., & Papadopoulos, M. C. (2006). Godfrey Hounsfield and the dawn of Computed Tomography. *Neurosurgery*, *58*(4), 780–787. https://doi.org/10.1227/01.neu.0000204309.91666.06

Powers, D. M. W. (2011). Evaluation: From Precision, Recall and F-Measure to ROC, Informedness, Markedness & Correlation. *Journal of Machine Learning Technologies*, *2*(1), 37–63.

Quinlan, J. R. (1986). Induction of Decision Trees. *Machine Learning*, *1*(1), 81–106. https://doi.org/10.1007/bf00116251

Rosenblatt, F. (1958). The Perceptron: A probabilistic model for information storage and organization in the brain. *Psychological Review*, *65*(6), 386–408. https://doi.org/10.1037/h0042519

Russakovsky, O., Deng, J., Su, H., Krause, J., Satheesh, S., Ma, S., Huang, Z., Karpathy, A., Khosla, A., Bernstein, M., Berg, A. C., & Fei-Fei, L. (2015). Imagenet Large Scale Visual Recognition Challenge. *International Journal of Computer Vision*, *115*(3), 211–252. https://doi.org/10.1007/s11263-015-0816-y

Sajja, T., Devarapalli, R., & Kalluri, H. (2019). Lung cancer detection based on CT scan images by using Deep Transfer Learning. *Traitement Du Signal*, *36*(4), 339–344. https://doi.org/10.18280/ts.360406

Samuel, A. L. (1959). Some studies in machine learning using the game of Checkers. *IBM Journal of Research and Development*, *3*(3), 210–229. https://doi.org/10.1147/rd.33.0210

Shivajirao Shinde, B. (2011). The origins of Digital Image Processing & application areas in Digital Image Processing Medical Images. *IOSR Journal of Engineering*, *1*(1), 66–71. https://doi.org/10.9790/3021-0116671

Silva, W. (2019). *Cnn-Pdm: A Convolutional Neural Network Framework For Assets Predictive Maintenance* (thesis). Western University .

Simonyan, K., & Zisserman, A. (2014). Very Deep Convolutional Networks for Large-Scale Image Recognition. *ArXiv Preprint ArXiv:1409.1556.*

Singh, G. A., & Gupta, P. K. (2018). Performance analysis of various machine learning-based approaches for detection and classification of lung cancer in humans. *Neural Computing and Applications*, *31*(10), 6863–6877. https://doi.org/10.1007/s00521-018-3518-x

Skalski, P. (2019, April 14). *Gentle dive into math behind Convolutional Neural Networks*. Medium. Retrieved January 1, 2022, from https://towardsdatascience.com/gentle-dive-into-math-behind-convolutional-neural-networks-79a07dd44cf9

Srinivas, S., Sarvadevabhatla, R. K., Mopuri, K. R., Prabhu, N., Kruthiventi, S. S., & Babu, R. V. (2016). A taxonomy of deep convolutional neural nets for Computer Vision. *Frontiers in Robotics and AI*, *2*. https://doi.org/10.3389/frobt.2015.00036

Subramanian, R. R., Mourya, R. N., Reddy, V. P. T., & Reddy, B. N. (2020). *International Journal of Control and Automation*, *13*(03), 154–160. https://doi.org/http://sersc.org/journals/index.php/IJCA/article/view/24979

Suganya, E., Sountharrajan, S., Shandilya, S. K., & Karthiga, M. (2019). IOT in agriculture investigation on plant diseases and nutrient level using image analysis techniques. *Internet of Things in Biomedical Engineering*, 117–130. https://doi.org/10.1016/b978-0-12-817356-5.00007-3

Szegedy, C., Wei Liu, Yangqing Jia, Sermanet, P., Reed, S., Anguelov, D., Erhan, D., Vanhoucke, V., & Rabinovich, A. (2015). Going deeper with convolutions. *2015 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*. https://doi.org/10.1109/cvpr.2015.7298594

Thakur, S. K., Singh, D. P., & Choudhary, J. (2020). Lung cancer identification: A review on detection and classification. *Cancer and Metastasis Reviews*, *39*(3), 989–998. https://doi.org/10.1007/s10555-020-09901-x

Toğaçar, M., Ergen, B., & Cömert, Z. (2020). Detection of lung cancer on chest CT images using minimum redundancy maximum relevance feature selection method with Convolutional Neural Networks. *Biocybernetics and Biomedical Engineering, 40*(1), 23–39. https://doi.org/10.1016/j.bbe.2019.11.004

Treloar, N. J., Fedorec, A. J., Ingalls, B., & Barnes, C. P. (2020). Deep reinforcement learning for the control of microbial co-cultures in bioreactors. *PLOS Computational Biology*, *16*(4). https://doi.org/10.1371/journal.pcbi.1007783

Turing, A. M. (1950). Computing Machinery and intelligence. *Mind, LIX*(236), 433–460. https://doi.org/10.1093/mind/lix.236.433

Wang, L. P., S., L., T., L., J., H. F., & D., W. (2020). A Large-Scale CT and PET/CT Dataset for Lung Cancer Diagnosis [Data set]. *The Cancer Imaging Archive*. https://doi.org/https://doi.org/10.7937/TCIA.2020.NNC2-0461

Wang, Y., Louie, D. C., Cai, J., Tchvialeva, L., Lui, H., Jane Wang, Z., & Lee, T. K. (2021). Deep learning enhances polarization speckle for in vivo skin cancer detection. *Optics & Laser Technology, 140,* 107006. https://doi.org/10.1016/j.optlastec.2021.107006

World Health Organization. (2021, September 21). *Cancer*. World Health Organization. Retrieved November 10, 2021, from https://www.who.int/news-room/fact-sheets/detail/cancer

Xu, C., & Jackson, S. A. (2019). Machine learning and complex biological data. *Genome Biology*, *20*(1). https://doi.org/10.1186/s13059-019-1689-0

Xu, Y., Fang, M., Chen, L., Du, Y., Tianyi Joey, Z., & C., Z. (2020). Deep Reinforcement Learning with Stacked Hierarchical Attention for Text-based Games. *ArXiv*. https://doi.org/https://arxiv.org/abs/2010.11655

Zagoruyko, S., & Komodakis, N. (2016). Wide residual networks. *Procedings of the British Machine Vision Conference 2016*. https://doi.org/10.5244/c.30.87

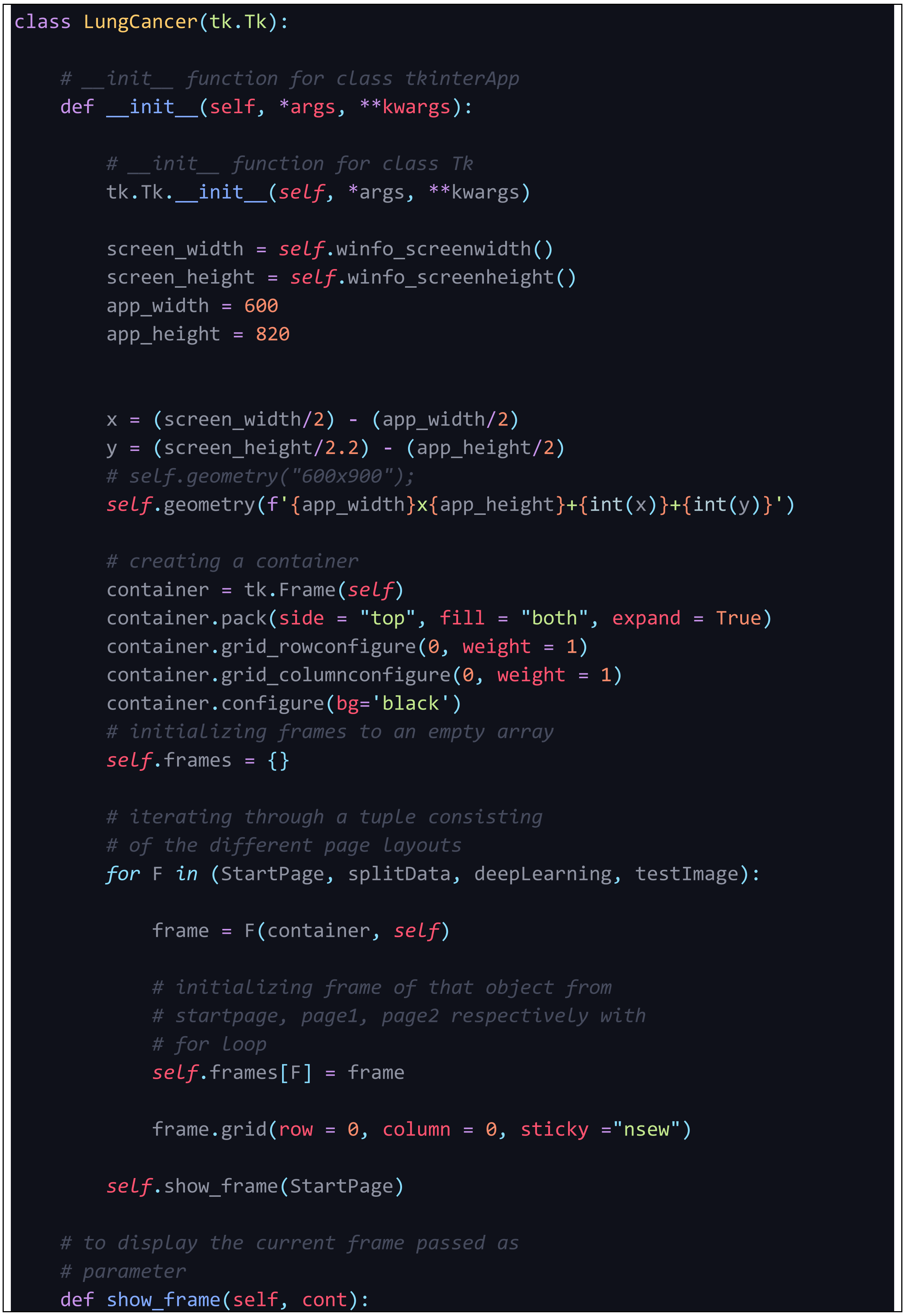
Zhai, X., Yin, Y., Pellegrino, J. W., Haudek, K. C., & Shi, L. (2020). Applying machine learning in science assessment: A systematic review. *Studies in Science Education*, *56*(1), 111–151. https://doi.org/10.1080/03057267.2020.1735757

Zhang, J., Chen, B., Zhou, M., Lan, H., & Gao, F. (2019). Photoacoustic image classification and segmentation of breast cancer: A feasibility study. *IEEE Access*, *7*, 5457–5466. https://doi.org/10.1109/access.2018.2888910

APPENDIX A: CODING FOR THE STUDY

app.py

```python
import copy
import csv
import logging
import math
import os
import queue
import shutil
import time
import tkinter as tk
from threading import *
from tkinter import (HORIZONTAL, Button, Entry, Frame, Label, StringVar,
                     filedialog)
from tkinter.scrolledtext import ScrolledText
from tkinter.ttk import Progressbar
import cv2
import matplotlib.pyplot as plt
import numpy as np
import pandas as pd
import torch
import torch.nn as nn
import torch.optim as optim
import torchvision
from bitarray import test
from matplotlib import pyplot as plt
from matplotlib import transforms
from PIL import Image, ImageTk
from sklearn import datasets
from sklearn.metrics import *
from torch.optim import lr_scheduler
from torchvision import datasets, models, transforms
import sugeno_integral

logger = logging.getLogger(__name__)
torch.manual_seed(8)
```



```
class LungCancer(tk.Tk):

    # __init__ function for class tkinterApp
    def __init__(self, *args, **kwargs):

        # __init__ function for class Tk
        tk.Tk.__init__(self, *args, **kwargs)

        screen_width = self.winfo_screenwidth()
        screen_height = self.winfo_screenheight()
        app_width = 600
        app_height = 820


        x = (screen_width/2) - (app_width/2)
        y = (screen_height/2.2) - (app_height/2)
        # self.geometry("600x900");
        self.geometry(f'{app_width}x{app_height}+{int(x)}+{int(y)}')

        # creating a container
        container = tk.Frame(self)
        container.pack(side = "top", fill = "both", expand = True)
        container.grid_rowconfigure(0, weight = 1)
        container.grid_columnconfigure(0, weight = 1)
        container.configure(bg='black')
        # initializing frames to an empty array
        self.frames = {}

        # iterating through a tuple consisting
        # of the different page layouts
        for F in (StartPage, splitData, deepLearning, testImage):

            frame = F(container, self)

            # initializing frame of that object from
            # startpage, page1, page2 respectively with
            # for loop
            self.frames[F] = frame

            frame.grid(row = 0, column = 0, sticky ="nsew")

        self.show_frame(StartPage)

    # to display the current frame passed as
    # parameter
    def show_frame(self, cont):
        frame = self.frames[cont]
        frame.tkraise()
        frame.configure(bg='black')

# first window frame startpage

class StartPage(tk.Frame):
    def __init__(self, parent, controller):

        tk.Frame.__init__(self, parent)

        mainLabel = tk.Label(self,text='Lung Cancer Detection: Lim Edition',
font=('Arial',25,'bold'),fg="white", bg="black",height=2).pack()

        img = ImageTk.PhotoImage(Image.open('lung.png'))
        label = tk.Label(self, image=img, bg='black')
        label.img = img

        splitDataButton = tk.Button(self, text ="Split Data", command =
lambda : controller.show_frame(splitData) ,borderwidth=0, width=50,
height=2, fg="pink", bg="black", font=("Helvetica", 18,'bold'))
        deepLearningButton = tk.Button(self, text ="Deep Learning", command
= lambda : controller.show_frame(deepLearning), borderwidth=0, width=50,
height=2, fg="pink", bg="black", font=("Helvetica", 18,'bold'))
        testImageButton = tk.Button(self, text ="Cancer Detection", command
= lambda : controller.show_frame(testImage), borderwidth=0, width=50,
height=2, fg="pink", bg="black", font=("Helvetica", 18,'bold'))
        quitButton = tk.Button(self, text="Exit",
command=self.quit,borderwidth=0, width=50, height=2, fg="pink", bg="black",
font=("Helvetica", 18,'bold'))

        label.pack(anchor='center')
        splitDataButton.pack(anchor='center', pady=(100,0))
        deepLearningButton.pack(anchor='center')
        testImageButton.pack(anchor='center')
        quitButton.pack(anchor='center')


# second window frame page1
class splitData(tk.Frame):

    def __init__(self, parent, controller):

        def inputFilesDir():
            self.inputFile = filedialog.askdirectory(title="Choose the 
location of the image")
            fileInput = os.path.abspath(self.inputFile)
            ROOT_DIR = fileInput
            number_of_images = {}

            if os.path.exists("./data/"):
                    shutil.rmtree("./data/")

            for dir in os.listdir(ROOT_DIR):
                number_of_images[dir] = 
len(os.listdir(os.path.join(ROOT_DIR, dir)))

            def dataFolder(p, split):

                os.makedirs("./data/"+p, exist_ok=True)

                for dir in os.listdir(ROOT_DIR):
                    os.mkdir('./data/'+p+'/' + dir)

                    for img in np.random.choice(a = 
os.listdir(os.path.join(ROOT_DIR, dir)), size = 
(math.floor(split*number_of_images[dir])-5), replace=False):
                        O = os.path.join(ROOT_DIR,dir,img)
                        D = os.path.join('./data/'+p,dir)
                        shutil.copy(O,D)


            dataFolder("train",0.8)
            dataFolder("val",0.2)

        def step():

            for i in range(5):
                self.update_idletasks()
                pb1['value'] += 30
                time.sleep(1)
            os.startfile(os.path.abspath('./data'))


        tk.Frame.__init__(self, parent)
        mainLabel = tk.Label(self,text='Split Data', 
font=('Arial',25,'bold'),fg="white", bg="black",height=2).pack()
        img = ImageTk.PhotoImage(Image.open('lung.png'))
        label = tk.Label(self, image=img, bg='black')
        label.img = img
        label.pack()
        pb1 = Progressbar(self, orient=HORIZONTAL, length=500)
        pb1.pack(pady=(100,0))

        # b1 = tk.Button(self, text="Choose
Folder",font=('Helvatica',16,'bold'),
command=lambda:[threading.Thread(target=inputFilesDir).start(),threading.Thr
ead(target=step).start()], height=2, width= 80, borderwidth=0, fg="pink",
bg="black")
        b1 = tk.Button(self, text="Choose
Folder",font=('Helvatica',16,'bold'),
command=lambda:[inputFilesDir(),step()], height=2, width= 80, borderwidth=0,
fg="pink", bg="black")
        b1.pack(pady=(20,0))
        b2 = tk.Button(self, text="Back To Main
Page",font=('Helvatica',15,'bold'),command=lambda :
controller.show_frame(StartPage),height=2, width= 80, borderwidth=0,
fg="pink", bg="black")
        b2.pack()

class QueueHandler(logging.Handler):
    """Class to send logging records to a queue
    It can be used from different threads
    The ConsoleUi class polls this queue to display records in a
ScrolledText widget
    """
    # Example from Moshe Kaplan:
https://gist.github.com/moshekaplan/c425f861de7bbf28ef06
    # (https://stackoverflow.com/questions/13318742/python-logging-to-
tkinter-text-widget) is not thread safe!
    # See https://stackoverflow.com/questions/43909849/tkinter-python-
crashes-on-new-thread-trying-to-log-on-main-thread

    def __init__(self, log_queue):
        super().__init__()
        self.log_queue = log_queue

    def emit(self, record):
        self.log_queue.put(record)

# third window frame page2
class deepLearning(tk.Frame):

    def display(self, record):
            msg = self.queue_handler.format(record)
            self.scrolled_text.configure(state='normal')
            self.scrolled_text.insert(tk.END, msg + '\n', record.levelname)
            self.scrolled_text.configure(state='disabled')

            # Autoscroll to the bottom
            self.scrolled_text.yview(tk.END)


    def poll_log_queue(self):
        # Check every 100ms if there is a new message in the queue to
display
            while True:
                try:
                    record = self.log_queue.get(block=False)
                except queue.Empty:
                    break
                else:
                    self.display(record)
            self.after(100, self.poll_log_queue)

    def __init__(self, parent, controller):

        def imshow(inp, title):
            """Imshow for Tensor."""
            inp = inp.numpy().transpose((1, 2, 0))
            mean = np.array([0.485, 0.456, 0.406])
            std = np.array([0.229, 0.224, 0.225])
            inp = std * inp + mean
            inp = np.clip(inp, 0, 1)
            plt.imshow(inp)
            plt.title(title)
            plt.show()

        def plot(val_loss,train_loss,typ):
            data_dir="./data/"
            plt.title("{} after epoch: {}".format(typ,len(train_loss)))
            plt.xlabel("Epoch")
            plt.ylabel(typ)
            plt.plot(list(range(len(train_loss))),train_loss,color="r",label
="Train "+typ)
            plt.plot(list(range(len(val_loss))),val_loss,color="b",label="Va
lidation "+typ)
            plt.legend()
            plt.savefig(os.path.join(data_dir,typ+".png"))
        #     plt.figure()
            plt.close()
        def train_model(model, criterion, optimizer,
scheduler,image_datasets,data_dir, num_epochs=25,model_name = "kaggle",):
            torch.manual_seed(8)
            data_dir="./data/"
            num_epochs=100
            device = torch.device("cuda:0" if torch.cuda.is_available() else
"cpu")
            val_loss_gph=[]
            train_loss_gph=[]
            val_acc_gph=[]
            train_acc_gph=[]
            since = time.time()

            best_model_wts = copy.deepcopy(model.state_dict())
            best_acc = 0.0

            for epoch in range(num_epochs):
                logger.log(logging.INFO, 'Epoch {}/{}'.format(epoch+1,
num_epochs))
                # print('Epoch {}/{}'.format(epoch+1, num_epochs))
                logger.log(logging.INFO, '-' * 10)
                # print('-' * 10)

                # Each epoch has a training and validation phase
                for phase in ['train', 'val']:
                    if phase == 'train':
                        model.train()  # Set model to training mode
                    else:
                        model.eval()   # Set model to evaluate mode

                    running_loss = 0.0
                    running_corrects = 0

                    # Iterate over data.
                    for inputs, labels in dataloaders[phase]:

                        inputs = inputs.to(device)
                        labels = labels.to(device)

                        # forward
                        # track history if only in train
                        with torch.set_grad_enabled(phase == 'train'):
                            outputs = model(inputs)
                            _, preds = torch.max(outputs, 1) #was
(outputs,1) for non-inception and (outputs.data,1) for inception
                            loss = criterion(outputs, labels)
                            # backward + optimize only if in training phase
                            if phase == 'train':
                                optimizer.zero_grad()
                                loss.backward()
                                optimizer.step()

                        # statistics
                        running_loss += loss.item() * inputs.size(0)
                        running_corrects += torch.sum(preds.detach() ==
labels.data)

                    if phase == 'train':
                        scheduler.step()

                    epoch_loss = running_loss / dataset_sizes[phase]
                    epoch_acc = running_corrects.double() /
dataset_sizes[phase]

                    if phase == 'train':
                        train_loss_gph.append(epoch_loss)
                        train_acc_gph.append(epoch_acc.item())
                    if phase == 'val':
                        val_loss_gph.append(epoch_loss)
                        val_acc_gph.append(epoch_acc.item())

                    plot(val_loss_gph,train_loss_gph, "Loss")
                    plot(val_acc_gph,train_acc_gph, "Accuracy")

                    logger.log(logging.INFO, '{} Loss: {:.4f} Acc:
{:.4f}'.format(
                        phase, epoch_loss, epoch_acc))
                    # print('{} Loss: {:.4f} Acc: {:.4f}'.format(
                    #     phase, epoch_loss, epoch_acc))

                    # deep copy the model
                    if phase == 'val' and epoch_acc >= best_acc:
                        best_acc = epoch_acc
                        best_model_wts = copy.deepcopy(model.state_dict())
                        torch.save(model.state_dict(),
data_dir+"/"+model_name+".pth")
                        logger.log(logging.INFO, '==>Model Saved')
                        # print('==>Model Saved')

                logger.log(logging.INFO, '')
                # print()
```

```
            time_elapsed = time.time() - since
            logger.log(logging.INFO, 'Training complete in {:.0f}m
{:.0f}s'.format(
                time_elapsed // 60, time_elapsed % 60))
            # print('Training complete in {:.0f}m {:.0f}s'.format(
                # time_elapsed // 60, time_elapsed % 60))
            logger.log(logging.INFO, 'Best val Acc: {:4f}'.format(best_acc))
            # print('Best val Acc: {:4f}'.format(best_acc))

            # load best model weights
            model.load_state_dict(best_model_wts)

            logger.log(logging.INFO, 'Getting the Probability Distribution')
            # print("\nGetting the Probability Distribution")
            testloader=torch.utils.data.DataLoader(image_datasets['val'],bat
ch_size=1)
            model=model.eval()

            correct = 0
            total = 0
            f = open(data_dir+'/'+model_name+".csv",'w+',newline = '')
            writer = csv.writer(f)

            with torch.no_grad():
                num = 0
                temp_array = np.zeros((len(testloader),num_classes))
                for data in testloader:
                    images, labels = data
                    labels=labels.cuda()
                    outputs = model(images.cuda())
                    _, predicted = torch.max(outputs, 1)
                    total += labels.size(0)
                    correct += (predicted == labels.cuda()).sum().item()
                    prob = torch.nn.functional.softmax(outputs, dim=1)
                    temp_array[num] =
np.asarray(prob[0].tolist()[0:num_classes])
                    num+=1
            # print("Accuracy = ",100*correct/total)
            logger.log(logging.INFO, "Accuracy = "+str(100*correct/total))
            # print("Accuracy = ",100*correct/total)

            for i in range(len(testloader)):
                writer.writerow(temp_array[i].tolist())
            f.close()
            return model

        # Getting Proba distribution
        # def get_probability(image_datasets,model,data_dir,model_name):
        #     data_dir="./data/"
        #     logger.log(logging.INFO, 'Getting the Probability
Distribution')
        #     # print("\nGetting the Probability Distribution")
        #     testloader=torch.utils.data.DataLoader(image_datasets['val'],b
atch_size=1)
        #     model=model.eval()

        #     correct = 0
        #     total = 0
        #     import csv

        #     import numpy as np
        #     f = open(data_dir+'/'+model_name+".csv",'w+',newline = '')
        #     writer = csv.writer(f)

        #     with torch.no_grad():
        #         num = 0
        #         temp_array = np.zeros((len(testloader),num_classes))
        #         for data in testloader:
        #             images, labels = data
        #             labels=labels.cuda()
        #             outputs = model(images.cuda())
        #             _, predicted = torch.max(outputs, 1)
        #             total += labels.size(0)
        #             correct += (predicted == labels.cuda()).sum().item()
        #             prob = torch.nn.functional.softmax(outputs, dim=1)
        #             temp_array[num] =
np.asarray(prob[0].tolist()[0:num_classes])
        #             num+=1
        #     print("Accuracy = ",100*correct/total)
        #     logger.log(logging.INFO, "Accuracy = ",100*correct/total)
        #     # print("Accuracy = ",100*correct/total)

        #     for i in range(len(testloader)):
        #         writer.writerow(temp_array[i].tolist())
        #     f.close()


        def learningPart(self, data_dir,num_epochs):
            mean = np.array([0.485, 0.456, 0.406])
            std = np.array([0.229, 0.224, 0.225])
            # data_dir="./data/"

            global data_transforms
            data_transforms = {
                'train': transforms.Compose([
                    transforms.Resize((224,224)),
                    transforms.ToTensor(),
                    transforms.Normalize(mean, std)
                ]),
                'val': transforms.Compose([
                    transforms.Resize((224,224)),
                    transforms.ToTensor(),
                    transforms.Normalize(mean, std)
                ]),
            }

            global image_datasets
            image_datasets = {x: datasets.ImageFolder(os.path.join(data_dir,
x),
                                                    data_transforms[x])
                        for x in ['train', 'val']}

            global dataloaders
            dataloaders = {x: torch.utils.data.DataLoader(image_datasets[x],
batch_size=4,
                                                        shuffle=True,
num_workers=4)
                        for x in ['train', 'val']}

            global dataset_sizes
            dataset_sizes = {x: len(image_datasets[x]) for x in ['train',
'val']}

            global class_names
            class_names = image_datasets['train'].classes

            global num_classes
            num_classes = len(class_names)

            global device
            device = torch.device("cuda:0" if torch.cuda.is_available() else
"cpu")
            # print(class_names)
            logger.log(logging.INFO, class_names)
```

```
        #Get probability distributions from the 4 models
            # global num_epochs
            num_epochs = 100

            global criterion
            criterion = nn.CrossEntropyLoss()

            # Get a batch of training data
            global inputs, classes
            inputs, classes = next(iter(dataloaders['train']))

            model = models.vgg11_bn(pretrained = True)
            optimizer = optim.SGD(model.parameters(), lr=0.0001, momentum =
0.99)
            step_lr_scheduler = lr_scheduler.StepLR(optimizer, step_size =
10, gamma=0.1)
            num_ftrs = model.classifier[0].in_features
            model.classifier = nn.Linear(num_ftrs, num_classes)
            model = model.to(device)
            logger.log(logging.INFO, '\nVGG-11')
            # model = train_model(model, criterion, optimizer,
step_lr_scheduler, num_epochs=num_epochs, model_name = 'Kaggle_vgg11')
            model = Thread(target=train_model(model, criterion, optimizer,
step_lr_scheduler,image_datasets,data_dir ,num_epochs=num_epochs, model_name
= 'Kaggle_vgg11')).start()
            #
get_probability(image_datasets,model,data_dir,model_name='Kaggle_vgg11')


            model = models.googlenet(pretrained = True)
            optimizer = optim.SGD(model.parameters(), lr=0.0001, momentum =
0.99)
            step_lr_scheduler = lr_scheduler.StepLR(optimizer, step_size =
10, gamma=0.1)
            num_ftrs = model.fc.in_features
            model.fc = nn.Linear(num_ftrs, num_classes)
            model = model.to(device)
            logger.log(logging.INFO, '\nGoogleNet')
            # model = train_model(model, criterion, optimizer,
step_lr_scheduler, num_epochs=num_epochs, model_name = 'Kaggle_googlenet')
            model = Thread(target=train_model(model, criterion, optimizer,
step_lr_scheduler,image_datasets,data_dir ,num_epochs=num_epochs, model_name
= 'Kaggle_googlenet')).start()
            #
get_probability(image_datasets,model,data_dir,model_name='Kaggle_googlenet')
```

```
            model = models.squeezenet1_1(pretrained = True)
            optimizer = optim.SGD(model.parameters(), lr=0.0001, momentum =
0.99)
            step_lr_scheduler = lr_scheduler.StepLR(optimizer, step_size =
10, gamma=0.1)
            model.classifier[1] = nn.Conv2d(512, num_classes,
kernel_size=(1,1), stride=(1,1))
            model = model.to(device)
            logger.log(logging.INFO, '\nSqueezeNet')
            # model = train_model(model, criterion, optimizer,
step_lr_scheduler, num_epochs=num_epochs, model_name = 'Kaggle_squeezenet')
            model = Thread(target=train_model(model, criterion, optimizer,
step_lr_scheduler,image_datasets,data_dir ,num_epochs=num_epochs, model_name
= 'Kaggle_squeezenet')).start()
            #
get_probability(image_datasets,model,data_dir,model_name='Kaggle_squeezenet'
)


            model = models.wide_resnet50_2(pretrained = True)
            optimizer = optim.SGD(model.parameters(), lr=0.0001, momentum =
0.99)
            step_lr_scheduler = lr_scheduler.StepLR(optimizer, step_size =
10, gamma=0.1)
            num_ftrs = model.fc.in_features
            model.fc = nn.Linear(num_ftrs, num_classes)
            model = model.to(device)
            logger.log(logging.INFO, '\nWideResNet-50-2')
            # model = train_model(model, criterion, optimizer,
step_lr_scheduler, num_epochs=num_epochs, model_name = 'Kaggle_wideresnet')
            model = Thread(target=train_model(model, criterion, optimizer,
step_lr_scheduler,image_datasets,data_dir ,num_epochs=num_epochs, model_name
= 'Kaggle_wideresnet')).start()
            #
get_probability(image_datasets,model,data_dir,model_name='Kaggle_wideresnet'
)


            prob1,labels = getfile("Kaggle_vgg11",root = data_dir)
            prob2,_ = getfile("Kaggle_squeezenet",root = data_dir)
            prob3,_ = getfile("Kaggle_googlenet",root = data_dir)
            prob4,_ = getfile("Kaggle_wideresnet",root = data_dir)

            ensemble_sugeno(labels,prob1,prob2,prob3,prob4)

        def threading():
            t1=Thread(target=learningPart(self,"./data/",100))
            t1.start()
```

```
    def getfile(filename, root="../"):
      file = root+filename+'.csv'
      df = pd.read_csv(file,header=None)
      df = np.asarray(df)

      labels=[]
      for i in range(19):
        labels.append(0)
      for i in range(107):
        labels.append(1)
      for i in range(78):
        labels.append(2)
      labels = np.asarray(labels)
      return df,labels

    def predicting(ensemble_prob):
      prediction = np.zeros((ensemble_prob.shape[0],))
      for i in range(ensemble_prob.shape[0]):
        temp = ensemble_prob[i]
        t = np.where(temp == np.max(temp))[0][0]
        prediction[i] = t
      return prediction

    def metrics(labels,predictions,classes):
      # print("Classification Report:")
      logger.log(logging.INFO, 'Classification Report:')
      # print(classification_report(labels, predictions, target_names
= classes,digits = 4))
      logger.log(logging.INFO, classification_report(labels,
predictions, target_names = classes,digits = 4))
      matrix = confusion_matrix(labels, predictions)
      tn=matrix[0][0]
      tp=matrix[1][1]
      fp=matrix[0][1]
      fn=matrix[1][0]
      print(tn)
      print(tp)
      print(fn)
      print(fp)
      # print("Confusion matrix:")
      logger.log(logging.INFO, 'Classification matrix:')
      # print(matrix)
      logger.log(logging.INFO, matrix)
      # print("\nClasswise
Accuracy :{}".format(matrix.diagonal()/matrix.sum(axis = 1)))
```

```
            logger.log(logging.INFO, "\nClasswise 
Accuracy :{}".format(matrix.diagonal()/matrix.sum(axis = 1)))
            # print("\nBalanced Accuracy Score: 
",balanced_accuracy_score(labels,predictions))
            logger.log(logging.INFO, "\nBalanced Accuracy Score: 
"+str(100*balanced_accuracy_score(labels,predictions)))


        #Sugeno Integral
        def ensemble_sugeno(labels,prob1,prob2,prob3,prob4):
            prob1.shape[1]
            Y = np.zeros(prob1.shape,dtype=float)
            for samples in range(prob1.shape[0]):
                for classes in range(prob1.shape[1]):
                    X = np.array([prob1[samples][classes], 
prob2[samples][classes], prob3[samples][classes], prob4[samples][classes] ])
                    measure = np.array([1.5, 1.5, 0.01, 1.2])
                    X_agg = 
sugeno_integral.sugeno_fuzzy_integral_generalized(X,measure)
                    Y[samples][classes] = X_agg

            sugeno_pred = predicting(Y)

            correct = np.where(sugeno_pred == labels)[0].shape[0]
            total = labels.shape[0]
            logger.log(logging.INFO, "Accuracy =" + str(correct/total))
            # print("Accuracy = ",correct/total)
            classes = ['Benign','Malignant','Normal']
            metrics(sugeno_pred,labels,classes)


        tk.Frame.__init__(self, parent)

        mainLabel = tk.Label(self,text='Deep Learning', 
font=('Arial',25,'bold'),fg="white", bg="black",height=2).pack()
        b1 = tk.Button(self, text="Start Deep Learning", command=
threading,borderwidth=0, width=50, height=3, fg="pink", bg="black", 
font=("Helvetica", 16,'bold'))
        b1.pack()

        self.scrolled_text=ScrolledText(self, state='disabled', height=35)
        self.log_queue = queue.Queue()
        self.queue_handler = QueueHandler(self.log_queue)
        formatter = logging.Formatter('%(message)s')
        self.queue_handler.setFormatter(formatter)
        logger.addHandler(self.queue_handler)
        # Start polling messages from the queue
        self.after(100, self.poll_log_queue)
        # T = tk.ScrolledText(self, height = 35, width = 70,
state='disable')
        self.scrolled_text.pack()

        b2 = tk.Button(self, text="Back To Main Page",command=lambda :
controller.show_frame(StartPage),borderwidth=0, width=50, height=3,
fg="pink", bg="black", font=("Helvetica", 16,'bold'))
        b2.pack()


class testImage(tk.Frame):
    def __init__(self, parent, controller):

        # open a image file from hard-disk
        def open_image(initialdir='/'):
            global file_path
            file_path  = filedialog.askopenfilename(initialdir=initialdir,
filetypes = [ ('Image File', '*.*' ) ]  )
            img_var.set(file_path)

            image = Image.open(file_path)
            image = image.resize((320,180)) # resize image to 32x32
            photo = ImageTk.PhotoImage(image)

            img_label = Label(middle_frame, image=photo, padx=10, pady=10)
            img_label.image = photo # keep a reference!
            img_label.grid(row=3, column=1)

            return file_path

        # ######################  Test Image
        def test_image():
            open_image()
            img = Image.open(file_path)
            mean = [0.485, 0.456, 0.406]
            std = [0.229, 0.224, 0.225]
            transform_norm = transforms.Compose([transforms.ToTensor(),
            transforms.Resize((224,224)),transforms.Normalize(mean, std)])
            # get normalized image
            img_normalized = transform_norm(img).float()
            img_normalized = img_normalized.unsqueeze_(0)
            # input = Variable(image_tensor)
            device = torch.device("cuda" if torch.cuda.is_available() else
"cpu")
```

```
            img_normalized = img_normalized.to(device)
            # print(img_normalized.shape)
            with torch.no_grad():
                model = torchvision.models.googlenet(pretrained=True)
                model.fc = torch.nn.Linear(in_features=1024, out_features=3,
bias=True)
                # print(model)
                model.load_state_dict(torch.load("extraFile\\Kaggle_googlene
t.pth"))

                model.state_dict()
                model.eval().cuda()
                output =model(img_normalized)
                index = output.data.cpu().numpy().argmax()
                train_ds = ['Benign','Malignant','Normal']
                global class_name
                class_name = train_ds[index]
                test_result_var.set("Test Result: " + class_name)
                return class_name


        """  Top Frame  """
        # tl = Label(top_frame, text="Top frame").pack()
        # ##### H5 #################
        tk.Frame.__init__(self, parent)

        top_frame = Frame(self, bd=10,bg="black")
        top_frame.pack()

        middle_frame = Frame(self,bd=10,bg="black")
        middle_frame.pack()

        bottom_frame = Frame(self, bd=10,bg="black")
        bottom_frame.pack()


        notification_frame = Frame(self,bd=10,bg="black")
        notification_frame.pack()

        mainLabel = tk.Label(top_frame,text='Cancer Detection',
font=('Arial',25,'bold'),fg="white", bg="black",height=2).grid(row=1,
column=1, columnspan=2)

        #######   IMAGE input
        btn_test = Button(top_frame, text='Press to Choose Image',  command
= test_image , font=('Helvetica',16,'bold'), height=2, borderwidth=0,
fg="pink", bg="black")
```

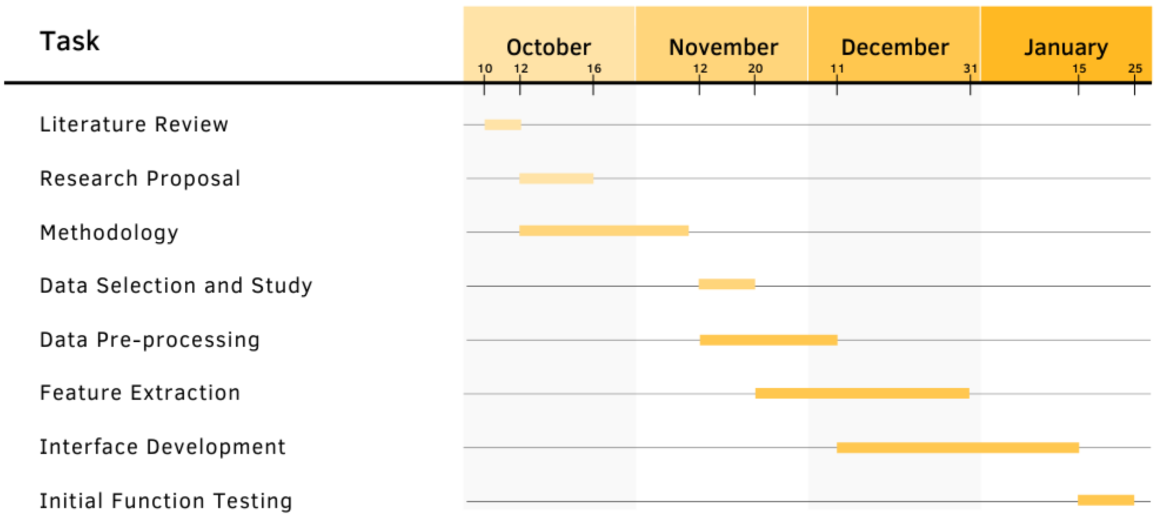
```
        btn_test.grid(row=2, column=1,columnspan=2, pady=(80,0))
        dir_input = Label(top_frame,font=("Helvetica",
15,'bold'),bg="black", fg="white", text="File Directory: ").grid(row=7,
column=1)
        img_var = StringVar()
        img_var.set("/")
        img_entry = Entry(top_frame, textvariable=img_var, width=60)

        img_entry.grid(row=7, column=2)

        """ middle Frame  """
        ml = Label(middle_frame, font=("Courier", 10),bg="black",
fg="white", text="Image Shown Below").grid(row=1, column=1)


        """ bottom Frame  """

        # Test Image butttom



        test_result_var = StringVar()
        test_result_var.set("Test Result:")
        test_result_label = Label(bottom_frame,
textvariable=test_result_var, font=('Helvetica',20,'bold'), height=2, width=
80, borderwidth=0, fg="white", bg="black").pack()
        b2 = tk.Button(self, text="Back To Main
Page",font=('Helvetica',16,'bold'),command=lambda :
controller.show_frame(StartPage),height=2, width= 80, borderwidth=0,
fg="pink", bg="black")
        b2.pack()


# Driver Code
if __name__ == '__main__':
    logging.basicConfig(level=logging.DEBUG)
    app = LungCancer()
    app.title('Lim LCD')
    app.mainloop()
```

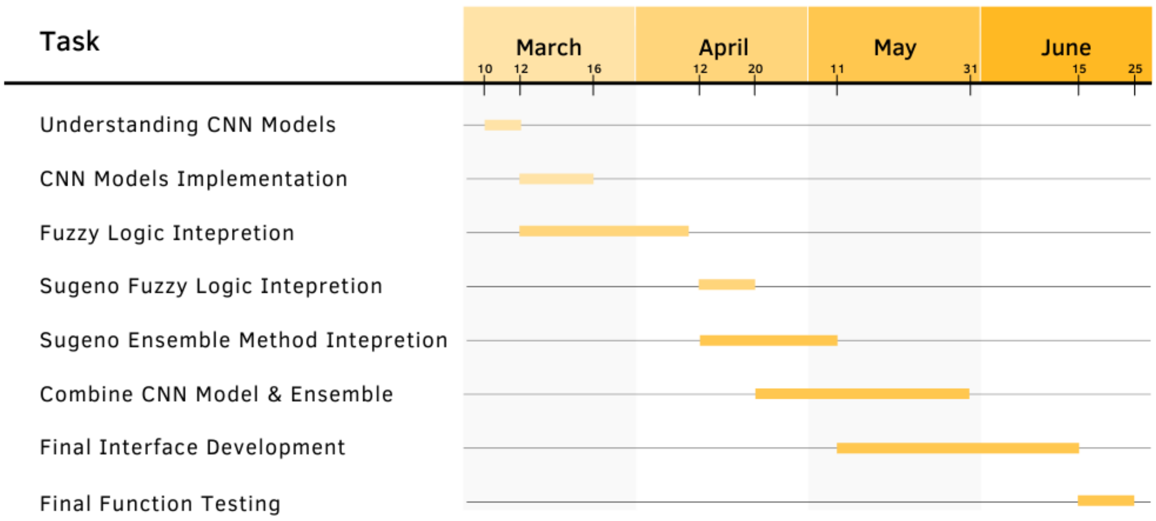
sugeno_integral.py

```
import numpy as np

```

```python
def generate_cardinality(N, p = 2):
    return [(x/ N)**p for x in np.arange(N, 0, -1)]


def sugeno_fuzzy_integral(X, measure=None, axis = 0, keepdims=True):
    if measure is None:
        measure = generate_cardinality(X.shape[axis])

    return sugeno_fuzzy_integral_generalized(X, measure, axis, np.minimum,
np.amax, keepdims)


def sugeno_fuzzy_integral_generalized(X, measure, axis = 0, f1 = np.minimum,
f2 = np.amax, keepdims=True):
    X_sorted = np.sort(X, axis = axis)
    return f2(f1(np.take(X_sorted, np.arange(0, X_sorted.shape[axis]),
axis), measure), axis=axis, keepdims=keepdims)
```

APPENDIX B: GANTT CHART


Task
October
November
December
January
Literature Review
Research Proposal
Methodology
Data Selection and Study
Data Pre-processing
Feature Extraction
Interface Development
Initial Function Testing


Task
March
April
May
June
Understanding CNN Models
CNN Models Implementation
Fuzzy Logic Intepretion
Sugeno Fuzzy Logic Intepretion
Sugeno Ensemble Method Intepretion
Combine CNN Model & Ensemble
Final Interface Development
Final Function Testing