

**CANCER DETECTION USING IMAGE PROCESSING
AND MACHINE/DEEP LEARNING METHODS**

LEONG ZEH ZEN

**A project report submitted in partial fulfilment of the
requirements for the award of the degree of
Bachelor of Engineering (Hons) Electronic Engineering**

**Faculty of Engineering and Green Technology
Universiti Tunku Abdul Rahman**

June 2022

DECLARATION

I hereby declare that this project report is based on my original work except for citations and quotations which have been duly acknowledged. I also declare that it has not been previously and concurrently submitted for any other degree or award at UTAR or other institutions.

Signature :  _____

Name : Leong Zeh Zen

ID No. : 18AGB00705

Date : 15 September 2022

APPROVAL FOR SUBMISSION

I certify that this project report entitled “**CANCER DETECTION USING IMAGE PROCESSING AND MACHINE/DEEP LEARNING METHODS**” was prepared by **LEONG ZEH ZEN** has met the required standard for submission in partial fulfilment of the requirements for the award of Bachelor of Engineering (Hons) Electronic Engineering at Universiti Tunku Abdul Rahman.

Approved by,

Signature : Humaira

Supervisor: Prof. Ts. Dr. Humaira Nisar

Date : 26 September 2022

The copyright of this report belongs to the author under the terms of the copyright Act 1987 as qualified by Intellectual Property Policy of Universiti Tunku Abdul Rahman. Due acknowledgement shall always be made of the use of any material contained in, or derived from, this report.

© 2022, LEONG ZEH ZEN. All right reserved.

ACKNOWLEDGEMENTS

I would like to thank everyone who had contributed to the successful completion of this project. I would like to express my gratitude to my research supervisor, Prof. Ts. Dr. Humaira Nisar, and my moderator, Ir. Dr. Chan Cheong Loong for their invaluable advice, guidance and their enormous patience throughout the development of the research.

In addition, I would also like to express my gratitude to my loving parent and friends who had helped and given me encouragement.

CANCER DETECTION USING IMAGE PROCESSING AND MACHINE/DEEP LEARNING METHODS

ABSTRACT

Breast cancer is one of the highest mortality cancers among women. The breast tumors can be classified into two categories, benign and malignant. Benign is the non-cancerous tumor; While the other variant, malignant is the cancerous tumor. These tumors are dangerous and mostly life-threatening due to the characteristics of the recurrence of the tumor. This is because the traditional classification methods are time-consuming, costly, labor-intensive and has reached their bottleneck. Integrating deep learning technology with medicinal solutions could improve the efficiency in early detection and treatment to improve the survival rates of breast cancer. Therefore, this paper researched the application of CNNs on the open-source Mendeley Breast Ultrasound dataset (MBU) by Rodrigues (2018) and the Breast Ultrasound Image dataset (BUSI) by Al-Dhabyani (2020). Moreover, the image pre-processing methods are implemented to refine the ultrasound image quality. Furthermore, the DCGAN model is used for data augmentation and to increase the data quantity. Subsequently, transfer learning-based approach is proposed for differentiating breast tumors. The proposed models, CNN-AlexNet, TL-Inception-V3 and TL-DenseNet are fine-tuned and trained on the MBU dataset. Moreover, the proposed classifier models are tested and evaluated on the BUSI dataset. The fine-tuned TL-DenseNet exhibited the finest performance among all proposed models by achieving an accuracy of 91.46% and F1-score of 0.9144, followed by the fine-tuned TL-Inception-V3 with accuracy of 91.04% and F1-score of 0.9100. The CNN-AlexNet also performs decently on the testing set with accuracy of 90.42% and F1-score of 0.9038.

TABLE OF CONTENTS

DECLARATION	ii
APPROVAL FOR SUBMISSION	iii
ACKNOWLEDGEMENTS	v
ABSTRACT	vi
TABLE OF CONTENTS	vii
LIST OF TABLES	x
LIST OF FIGURES	xii
LIST OF SYMBOLS / ABBREVIATIONS	xvii
LIST OF APPENDICES	xviii

CHAPTER

1	INTRODUCTION	1
	1.1 Background	1
	1.2 Problem Statements	5
	1.3 Project Scope	6
	1.4 Project Objectives	7
2	LITERATURE REVIEW	8
	2.1 Overview	8
	2.2 Generative Adversarial Network (GANs)	8
	2.3 Deep Convolutional Generative Adversarial Networks (DCGANs)	10
	2.4 GoogleLeNet (Inception)	11
	2.4.1 Inception-v1	12

	2.4.2	Inception-v2 & Inception-v3	13
	2.5	Residual Neural Network (ResNet)	16
	2.6	Related Works	20
3		METHODOLOGY	30
	3.1	Overview	30
	3.2	Environment Setup	31
		3.2.1 Hardware	31
		3.2.2 Software	32
	3.3	Data Processing	32
		3.3.1 Dataset Preparation	32
		3.3.2 Image Pre-Processing	34
		3.3.3 Image Augmentation	35
		3.3.4 Data Augmentation	36
		3.3.5 Data Segmentation	38
	3.4	Classification Model	39
		3.4.1 Dataset Cross-Validation	39
		3.4.2 CNNs and TL Architecture Design	40
	3.5	Evaluation Method	45
	3.6	Project Timeline	46
4		RESULTS AND DISCUSSIONS	48
	4.1	Overview	48
	4.2	Image Pre-Processing	49
	4.3	Image Augmentation	50
	4.4	Data Augmentation using DCGAN	51
	4.5	Training Results	53
		4.5.1 CNN-AlexNet	54
		4.5.2 TL-Inception-V3 with 3 extra hidden layers + dropout	58
		4.5.3 TL-DenseNet with 6 extra hidden layers + dropout	62
	4.6	Testing Results on BUSI dataset	66

		ix
4.7	Comparison between Existing Techniques	67
4.8	Discussion	69
5	CONCLUSION AND RECOMMENDATIONS	74
5.1	Project Review	74
5.2	Project Findings	75
5.3	Recommendations for Future Improvement	76
5.4	Conclusion	77
	REFERENCES	78
	APPENDICES	83

LIST OF TABLES

TABLE	TITLE	PAGE
	Table 1: Architecture details of Inception -v1 (Szegedy et al., 2015)	13
	Table 2: Proposed network architecture of Inception-v2 (Szegedy et al., 2016)	16
	Table 3: Top-1 error on ImageNet Validation (He et al., 2016)	18
	Table 4: Disc score evaluation in terms of mean and standard deviation of GAN-based model and BRATS'17 best model (Wang et al.). The GAN-based models were trained with augmentation and without augmentation (Shin et al., 2018)	20
	Table 5: Comparison of different DL and ML classifier in terms of accuracy on unprocessed data and CNN denoised data (Latif et al., 2019)	26
	Table 6: Classification results of different machine learning classifier with or without BGWO feature selection process in terms of accuracy and AUC (Khanna et al., 2021)	27
	Table 7: Classification results of VGG16, VGG19, Inception-V3 and SqueezeNet integrated with different ML algorithm on breast tumour classification (Gupta et al., 2022)	28
	Table 8: Summary of performance from various related papers	29
	Table 9: Hardware details	31
	Table 10: Python libraries version	32
	Table 11: Smoothing filter implemented in this project	35
	Table 12: Configuration used in CNN-AlexNet	40

Table 13: Terminology of Confusion Matrix	45
Table 14: Project Gantt Chart	47
Table 15: Pre-processed filtered images	49
Table 16: Samples of augmented image	50
Table 17: Samples of synthesized image generated by DCGANs	51
Table 18: Number of images in the dataset	52
Table 19: Comparison between proposed model in terms of accuracy, loss, precision, recall and F1-score on the validation dataset	53
Table 20: Validation Evaluation Metrics for CNN-AlexNet in Each Fold	54
Table 21: Validation Evaluation Metrics for TL-Inception-V3 with 3 extra hidden layers + dropout in Each Fold	58
Table 22: Validation Evaluation Metrics for TL-DenseNet with 6 extra hidden layers + dropout in Each Fold	62
Table 23: Comparison between proposed model in terms of accuracy, precision, recall and F1-score on the BUSI testing dataset	67
Table 24: Comparison between proposed models and existing techniques	68
Table 25: Samples of DCGANs synthesized image with and without batch normalisation layer at 700 training epochs	72

LIST OF FIGURES

FIGURE	TITLE	PAGE
	Figure 1-1: Incidence of deaths worldwide of different diseases in 2019 (Our World in Data, 2019)	2
	Figure 1-2: Incidence of cancer worldwide in 2020 (Global Cancer Observatory, 2020)	3
	Figure 1-3: Mortality rate of different cancers worldwide in 2020 (Global Cancer Observatory, 2020)	3
	Figure 1-4: Example of 3x3 kernel applied on an image (GeeksforGeeks, 2022)	4
	Figure 2-1: Simplified GANs framework (GeeksforGeeks, 2022)	9
	Figure 2-2: Evolution of GANs (Brownlee, 2019)	10
	Figure 2-3: Structure of DCGANs (Radford, 2016)	10
	Figure 2-4: Naïve version of Inception-v1 module (Szegedy et al., 2015)	12
	Figure 2-5: Dimension reductions inception-v1 module (Szegedy et al., 2015)	12
	Figure 2-6: The 5×5 convolution has been replaced by two 3×3 convolution in Inception-v2 (Szegedy et al., 2016)	14
	Figure 2-7: Factorization in Inception-v2 module (Szegedy et al., 2016)	15
	Figure 2-8: Filter banks outputs of Inception-v2 module were expanded (Szegedy et al., 2016)	15
	Figure 2-9: ResNet Residual Blocks (He et al., 2016)	17
	Figure 2-10: Refined residual blocks (Fung, 2017)	18

Figure 2-11: Samples of network architecture. Left: VGG-19 model. Mid: plain network inspired by VGG nets of 34 layers. Right: residual network of 34 layers (He et al., 2016)	19
Figure 2-12: Generator structure of MIGAN (Iqbal et al., 2018)	21
Figure 2-13: Discriminator structure of MIGAN (Iqbal et al., 2018)	21
Figure 2-14: Examples of two different input and their respective generated output (Senaras et al., 2018)	22
Figure 2-15: Original and GANs generated images with batch size 4 and 32 (Desai et al., 2020)	23
Figure 2-16: Accuracy of deep learning breast cancer classification (Desai et al., 2020)	24
Figure 2-17: CNN predictions in terms of F1-score with four approaches: original input (Blue), augmented original input (Orange), GANs input (Green) and augmented GANs input (Red) (Alyafi et al., 2020)	25
Figure 2-18: Comparison of performance in terms of accuracy on different dataset augmentation methods and CNNs architecture (Al-Dhabyani et al., 2019)	26
Figure 3-1: Project Methodology	31
Figure 3-2: Samples of benign tumour images of MBU dataset	33
Figure 3-3: Samples of malignant tumour images of MBU dataset	33
Figure 3-4: Samples of benign tumour images of BUSI dataset	33
Figure 3-5: Samples of malignant tumour images of BUSI dataset	33
Figure 3-6: Visualization of dataset of the synthesized data and original data	37
Figure 3-7: Visualization of dataset distribution	38
Figure 3-8: Terminology of 5-fold cross validation (Kumar, 2022)	39
Figure 3-9: Model summary of the CNN-AlexNet architecture implemented	42

Figure 3-10: Model summary of the TL-Inception-V3 architecture implemented	43
Figure 3-11: Model summary of the TL-DenseNet architecture implemented	44
Figure 4-1: Bar chart of comparison between proposed model on validation dataset	53
Figure 4-2: Training and Validation Accuracy against Number of Epochs for CNN-AlexNet in (a) First Fold (b) Second Fold (c) Third Fold (d) Fourth Fold (e) Fifth Fold	55
Figure 4-3: Training and Validation Loss against Number of Epochs for CNN-AlexNet in (a) First Fold (b) Second Fold (c) Third Fold (d) Fourth Fold (e) Fifth Fold	56
Figure 4-4: Confusion Matrix graph for CNN-AlexNet in (a) First Fold (b) Second Fold (c) Third Fold (d) Fourth Fold (e) Fifth Fold	57
Figure 4-5: Training and Validation Accuracy against Number of Epochs for TL-Inception-V3 with 3 extra hidden layers + dropout in (a) First Fold (b) Second Fold (c) Third Fold (d) Fourth Fold (e) Fifth Fold	59
Figure 4-6: Training and Validation Loss against Number of Epochs for TL-Inception-V3 with 3 extra hidden layers + dropout in (a) First Fold (b) Second Fold (c) Third Fold (d) Fourth Fold (e) Fifth Fold	60
Figure 4-7: Confusion Matrix graph for TL-Inception-V3 with 3 extra hidden layers + dropout in (a) First Fold (b) Second Fold (c) Third Fold (d) Fourth Fold (e) Fifth Fold	61
Figure 4-8: Training and Validation Accuracy against Number of Epochs for TL-DenseNet with 6 extra hidden layers + dropout in (a) First Fold (b) Second Fold (c) Third Fold (d) Fourth Fold (e) Fifth Fold	63
Figure 4-9: Training and Validation Loss against Number of Epochs for TL-DenseNet with 6 extra hidden layers + dropout in (a) First Fold (b) Second Fold (c) Third Fold (d) Fourth Fold (e) Fifth Fold	64
Figure 4-10: Confusion Matrix graph for TL- DenseNet with 6 extra hidden layers + dropout in (a) First Fold (b)	

Second Fold (c) Third Fold (d) Fourth Fold (e) Fifth Fold	65
Figure 4-11: Confusion Matrix Graph for (a) CNN-AlexNet (b) TL-Inception-V3 (c) TL-DenseNet on the BUSI dataset	66
Figure 4-12: Bar chart of comparison between proposed model on BUSI testing set	67
Figure 4-13: Bar chart of comparison between proposed model and existing techniques in terms of accuracy	69
Figure 4-14: Sample distribution of the dataset before and after applying DCGAN	71
Figure 4-15: Output statement of (a) TL-DenseNet and (b) TL- Inception-V3 before fine-tuned	73
Figure 5-1: Outline of the implementation of Pix2Pix technique to generate lung cancer CT image (Toda et al., 2022)	77
Figure 5-2: Training output statement of CNN-AlexNet in fold 1	83
Figure 5-3: Training output statement of CNN-AlexNet in fold 2	83
Figure 5-4: Training output statement of CNN-AlexNet in fold 3	84
Figure 5-5: Training output statement of CNN-AlexNet in fold 4	84
Figure 5-6: Training output statement of CNN-AlexNet in fold 5	84
Figure 5-7: Training output statement of TL-Inception-V3 in fold 1	85
Figure 5-8: Training output statement of TL-Inception-V3 in fold 2	85
Figure 5-9: Training output statement of TL-Inception-V3 in fold 3	85
Figure 5-10: Training output statement of TL-Inception-V3 in fold 4	86
Figure 5-11: Training output statement of TL-Inception-V3 in fold 5	86
Figure 5-12: Training output statement of TL-DenseNet in fold 1	86

Figure 5-13: Training output statement of TL-DenseNet in fold 2	87
Figure 5-14: Training output statement of TL-DenseNet in fold 3	87
Figure 5-15: Training output statement of TL-DenseNet in fold 4	87
Figure 5-16: Training output statement of TL-DenseNet in fold 5	88
Figure 5-17: Testing output statement of CNN-AlexNet	88
Figure 5-18: Testing output statement of TL-Inception-V3	88
Figure 5-19: Testing output statement of TL-DenseNet	88

LIST OF SYMBOLS / ABBREVIATIONS

Adaboost	Adaptive Boosting
ADAM	Adaptive Moment Estimation
AUC	Area Under the Curve
BGWO	binary Grey Wolf Optimization
CNN	Convolutional Neural Network
CT	Computed Tomography
CPU	Central Processing Unit
DCGANs	Deep Convolutional Generative Adversarial Network
DenseNet	Dense Convolutional Network
eLU	Exponential Linear Unit
GANs	Generative Adversarial Network
GPU	Graphics Processing Unit
ILSVRC	ImageNet Large Scale Visual Recognition Challenge
KNN	K-Nearest Neighbour
LeakyReLU	Leaky Rectified Linear Unit
LR	Logistic Regression
MRI	Magnetic Resonance Imaging
MP	Multilayer Perceptron
NN	Neural Networks
OS	Operating System
RAM	Random-Access Memory
ReLU	Rectified Linear Unit
ResNet	Residual Neural Network
RF	Random Forest
SVM	Support Vector Machine
TL	Transfer Learning

LIST OF APPENDICES

APPENDIX	TITLE	PAGE
APPENDIX A:	Training Output Statement of Proposed Models Generated in PyCharm	83
APPENDIX B:	Computer Programme Listing	89

CHAPTER 1

INTRODUCTION

1.1 Background

Cancer is defined as the condition of any disease which are characterized by abnormal cells duplicating and spreading uncontrollably to other organs of the body. The occurrence of cancer is due to the cell division process of our human body. Cell division occurs when damaged cells died, new cells would duplicate and replace the old cell. However, sometimes this process does not work orderly, this is where abnormal cells start to develop and duplicate uncontrollably. The duplication of abnormal cells creates a lump of solid tissue called a tumor. The tumor is also known as a neoplasm could affect skin, organs, and bones (National Cancer Institute, 2021).

Tumor can be classified into two categories, which are benign and malignant. This is because some tumors would not affect another tissue and it is not cancerous. The non-cancerous tumor also known as a benign tumor, and this category of tumor is impossible to spread the cancerous cells to nearby tissue. Moreover, most benign tumor could self-recover after some time without any medical treatment. However, there is currently no research on the transformation of a tumor (MedicalNewsToday, 2020), thus it is possible for a benign tumor turns into a malignant tumor, hence patients are suggested to seek for professional medical treatment if a benign tumor is detected. On the other hand, malignant tumors are known as the cancerous tumor, it could duplicate and spread the tissue build up by abnormal cells to nearby tissues and even other parts of the human's body. These tumors are dangerous and mostly life-threatening due to the characteristics of the recurrence of the tumor. There is a

probability of the cancerous tumor returning even if the tumor is treated beforehand. (Cleveland Clinic, 2021).

Every individual should pay high attention to cancer since it has the second highest mortality rate in the world standing at 10.08 million in 2019; while cardiovascular disease has the highest mortality rate (Our World in Data, 2019).

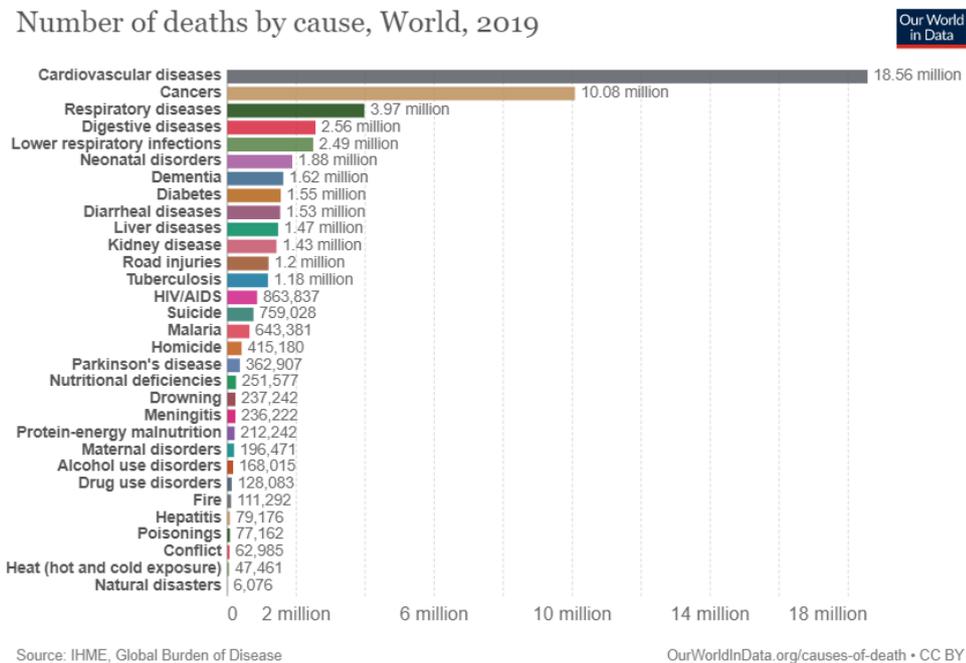


Figure 1-1: Incidence of deaths worldwide of different diseases in 2019 (Our World in Data, 2019)

Furthermore, the data from Global Cancer Observatory shows that breast cancer has the highest incidence in 2020, with a number of occurrences of 11.7% of the total cancer incidence recorded in 2020, which means among 19 million of cancer diagnosed in 2020, there are 2.26 million of the total cases are breast cancer. Apart from that, breast cancer has a mortality rate of 6.9% among all different types of cancers, which means among 10 million cancer-related death cases in 2020, there are 684 thousand cases are breast cancer-related patients.

Estimated number of new cases in 2020, worldwide, both sexes, all ages

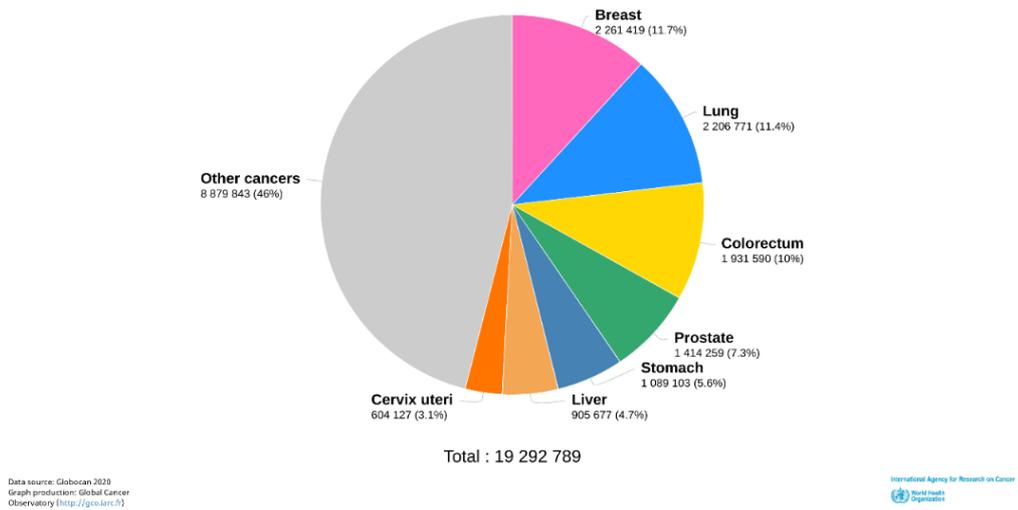


Figure 1-2: Incidence of cancer worldwide in 2020 (Global Cancer Observatory, 2020)

Estimated number of deaths in 2020, worldwide, both sexes, all ages

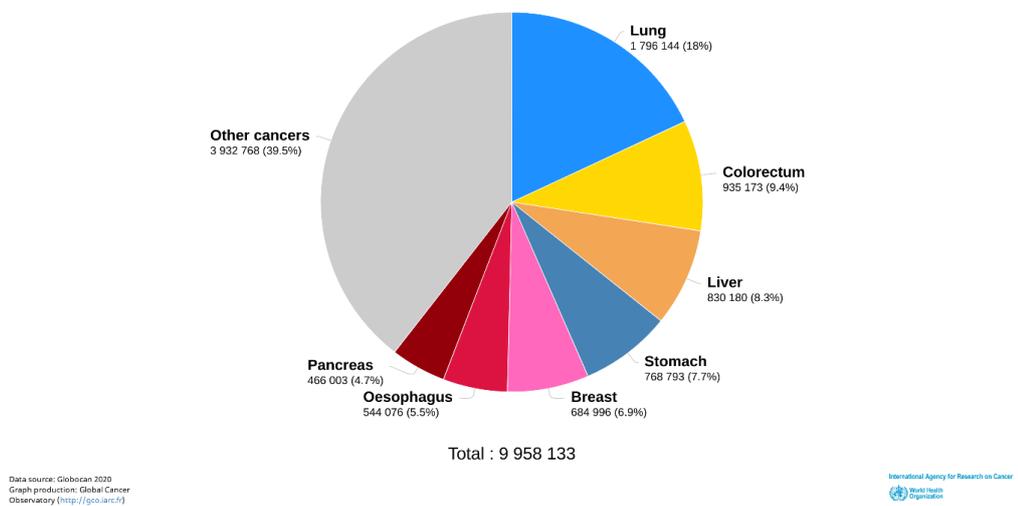


Figure 1-3: Mortality rate of different cancers worldwide in 2020 (Global Cancer Observatory, 2020)

Since breast cancer is life-threatening and it is also one of the leading causes of death, therefore early diagnosis acts as an important role in order to prevent cancer from progressing rapidly and starting to affect human health condition or even worse, approaching death. Moving along with the improvement of technology and innovation, breast cancer screening methods for instance Mammography, Magnetic

Resonance Imaging (MRI), ultrasound scanning, etc. are getting more advance and mature. For example, ultrasound scanning emits sound waves with a frequency of 7.5MHz to 13MHz to image the internal structure of our body (Kuhl et al., 2005).

Due to the noises that occurs in the ultrasound images, some image processing methods can be applied as a solution to overcome this problem. Medical image processing is the practice of enhancing the medical image by reducing the image noise and easing the interpretation by both humans and machines. Medical images are made up of pixels which is the smallest element of an image. Each pixel represents a single numeric value therefore different pixels with different numeric values illustrate as different colors in a single image.

One of the most important techniques in image processing is convolution. Convolution is defined as a process by applying a kernel to each pixel and its nearby pixels over the whole image, hence transforming the image. The impact of the convolution process's transformation is determined by the size and values of the kernel, which is a matrix of values (Basavarajaiah, 2022). Moreover, the medical image can be improved by applying a kernel that act as a smoothing mask over a convolution to achieve the effect of blurring the image by reducing the image noise and smoothening the edges.

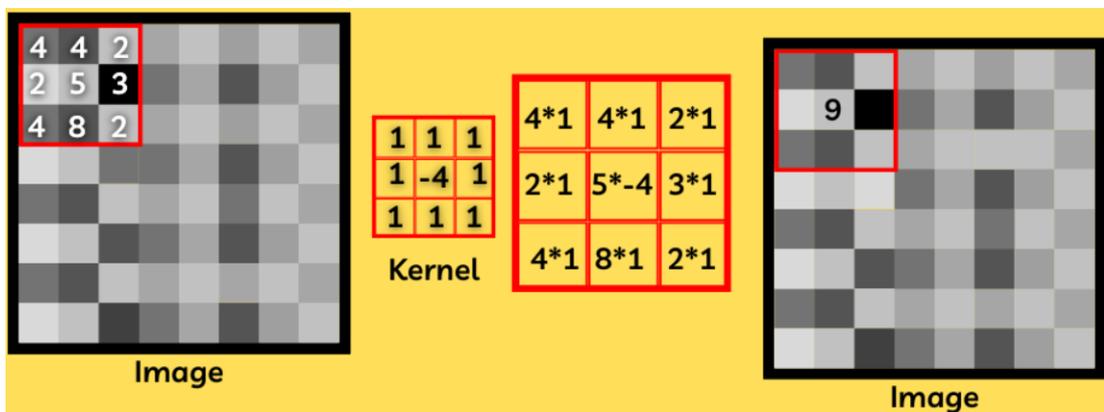


Figure 1-4: Example of 3x3 kernel applied on an image (GeeksforGeeks, 2022)

Despite the recent developments in breast cancer screening methods, experience pathologists' visual inspections are still crucial for diagnosing breast

cancer correctly. Moreover, the diagnosis results are arbitrary and could subject to be different diagnosis results depending on the observations. Apart from that, the diagnosis process can be highly time-consuming and difficult. Besides, since the process can be tedious, therefore it might lead to misdiagnosis of the pathologists' visual inspections. Hence, building automate computer systems such as image processing and deep learning model could improve the efficiency of pathologists by reducing their workload. Besides, automate computer systems also benefits in reducing the subjectivity of the breast cancer classification (Zhi et al., 2017).

However, the deep learning models required a sizeable amount of dataset in order to obtain higher classification accuracy. Regrettably, because of privacy concerns and the expensive expense of expert annotations, publicly available medical-related datasets are generally small and skewed. Therefore, the Generative adversarial networks (GANs) introduced by Goodfellow et al. in 2014 could greatly restrict the drawbacks of small datasets by generating synthetic medical images based on the available datasets. With the ability of GANs to replicate data distributions and synthesize images has successfully paved the way for new techniques to overcome the drawbacks of both supervised deep learning and generating synthetic images (Kazeminia et al., 2020).

1.2 Problem Statements

Breast cancer is one of the highest incidence cancers in Malaysia. According to the Malaysia National Cancer Registry 2004, the ASR of breast cancer in the country is 46.2 over 100,000 women. Furthermore, the overall 5-year survival rate for breast cancer patients in Malaysia is 49% with a median survival period of 68.1 months (Yip et al., 2006).

In recent years, one of the most frequently used imaging technologies in clinical practice is ultrasound imaging. The ultrasound imaging method is considered a dynamically developing technology with numerous advantages and it has been acknowledged as a potent and commonplace screening and diagnostic tool for

clinical research practice. Especially, due to its overall reasonable cost, operator expertise, and relatively lower impact on human health. Besides, the ultrasound imaging technology has been widely implemented in the fields of breast diagnostics. Nevertheless, the ultrasound imaging technology also comes with several major drawbacks, for instance, acquisition noises generated by the ultrasound imaging machine, ambient noises generated by the surroundings of the ultrasound imaging took place, and the presence of body fat, organs, and other tissues could significantly affect the image quality (Hiremath et al., 2013).

Moreover, the traditional breast cancer diagnostics methods are mostly costly, time consuming, and relied on the extensive experience of the diagnostician and specialists. Apart from that, the availability of open access breast tumor datasets is very less due to the privacy of the patients.

1.3 Project Scope

The project aims at developing a deep neural network that could import the breast cancer ultrasound images from the Mendeley website and classify the tumor types into two classes, which are benign and malignant. Furthermore, this project proposes implementing the Generative Adversarial Networks (GANs) model to synthesize both benign and malignant realistic breast tumor images to solve the problem of lacking data and augment the skewed datasets that could cause classification problems. Moreover, image processing methods such as image filtering and image smoothing are implemented to deblur and reduce the noises in the ultrasound images in order to improve the deep neural network model training quality. Lastly, a deep learning classifier algorithm should be developed in order to classify the benign and malignant ultrasound images.

1.4 Project Objectives

The objectives of the project are shown below:

- i) Design an image processing method to reduce the image noise of the ultrasound images of benign and malignant tumors.
- ii) Application of data augmentation, such as GAN model to increase the dataset quantity and improve the quality of the dataset.
- iii) Design a CNN model to classify benign and malignant tumors.
- iv) Apply a suitable transfer learning model and fine-tuned the model's parameters in order to improve the accuracy of the classifier.

CHAPTER 2

LITERATURE REVIEW

2.1 Overview

This chapter aims to review the data augmentation method, such as the Generative Adversarial Networks (GAN). Besides, the framework for Convolutional Neural Networks (CNN) will be discussed in this chapter. Furthermore, this chapter also reviewed and discussed previous research papers related to the project.

2.2 Generative Adversarial Network (GANs)

The Generative Adversarial Network also known as GANs is proposed by Goodfellow et. al. in 2014. The Generative Adversarial Network was proposed to estimate the generative models by using an adversarial process. The framework includes two different models, which are the generative model and a discriminative model. The generative model is trained to capture the data distribution and generate new examples from the training data. On the other hand, the discriminative model is responsible to estimate the probability of the samples whether it is generated by the generative model or from the training data.

In the advancement of artificial intelligence (AI), the discriminative model has been developed with great success and is widely used in major machine learning models. The algorithm behind the majority of the discriminative model is based on a

backpropagation algorithm with a suitable gradient. However, the generative model has issues facing difficulties in approximating multiple interactable probabilistic calculations that emerge in maximum likelihood estimation and related methodologies. Therefore, the proposed generative model is to overcome the drawbacks.

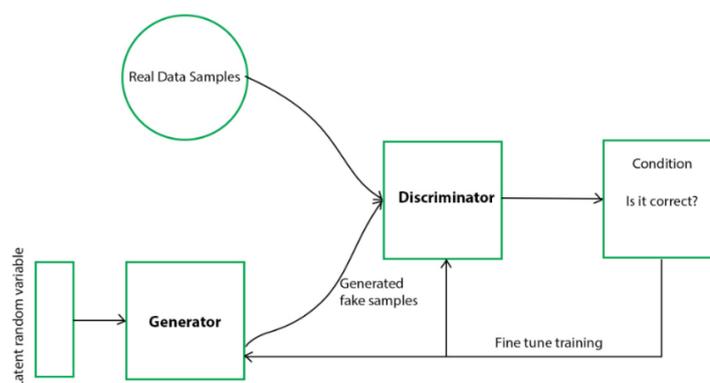


Figure 2-1: Simplified GANs framework (GeeksforGeeks, 2022)

The Generative Adversarial Network framework is proposed to design the generative model with an adversary. The main motivation of the generative model is to generate counterfeit samples until it is indistinguishable from the actual training data. While the discriminative model of the GANs framework is trained to predict the probability of the sample from the data distribution and model distribution, which is exactly the sample generated by the generative model. In order to improve the discriminative model, the generative model is responsible to worsen the estimation of the discriminative model. To summarize concisely, the objective of the generative model is achieved when the discriminative model is facing difficulties in classifying the samples. Hence, the generative model and discriminative model are both adversaries to each other. Furthermore, with the adversary of the discriminative model, the generative model can generate indistinguishable counterfeit samples from any random input without interfering with the actual training data. Apart from that, both generative and discriminative models are multilayer perceptrons. The GANs framework is trained with backpropagation and dropout algorithms from the

discriminative predictions and the generative model is trained with forward propagation only (Goodfellow et al., 2014).

Over the past few years, GANs have been widely researched and developed because of the proven successful framework. At present, GANs is capable to generate images so realistic that it is difficult to identify the counterfeit.



Figure 2-2: Evolution of GANs (Brownlee, 2019)

2.3 Deep Convolutional Generative Adversarial Networks (DCGANs)

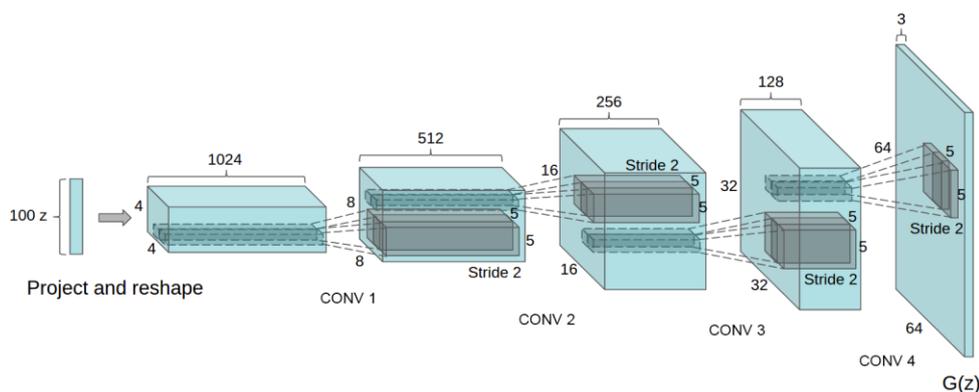


Figure 2-3: Structure of DCGANs (Radford, 2016)

The Deep Convolutional Generative Adversarial Network also known as DCGAN was introduced by Radford et al. in 2015. In the past few years, supervised learning with CNNs has been advanced and utilized in multiple computer vision applications, but in comparison, unsupervised learning with CNNs is less developed and adopted.

Therefore, DCGAN is proposed to enhance the development of unsupervised learning with CNNs (Radford et al, 2016).

The structure of the DCGAN is inspired by the Improved GANs introduced by Salimans, Goodfellow, Zaremba, et al. in 2016. The Improved GANs defined three enhancement techniques which are feature matching, minibatch discrimination, and historical averaging to stabilize the training model. These techniques improve the variety of the discriminate network by improving the diversity of samples created by the generative model when discriminating samples. With the inspiration of the improved GANs, therefore DCGAN has expanded GAN from multilayer perceptron (MLP) structure into convolutional neural network (CNN) structure (Fang et al., 2018).

According to Radford et al., DCGAN has achieved an impressive result on real datasets, such as LSUN and CelebA. Furthermore, there are a few modifications in the integration of the architecture of GANs and CNNs to stabilize the DCGANs.

- i) The CNNs max pooling layers are replaced with the strided convolutions also known as a discriminator to learn the network spatial down sampling and fractional-strided aka generators to learn the network spatial up sampling.
- ii) The fully connected hidden layers are eliminated.
- iii) Batch Normalization is applied in the discriminator and generator model.
- iv) Generator – ReLU activation for all layers; tanh for output layer.
- v) Discriminator – LeakyReLU for all layers (Radford et al, 2016).

2.4 GoogleLeNet (Inception)

The GoogleLeNet microarchitecture also known as Inception is one of the most used deep CNN architecture in deep learning. The architecture was first introduced by Szegedy et al. in 2014. Besides, it is also the winner of ILSVRC14 with an error rate of 6.67%, and has significantly outperform the previous ILSVRC winner, AlexNet (ILSVRC13 winner) and ZFNet (ILSVRC12 winner).

2.4.1 Inception-v1

The first version of GoogleLeNet aka Inception-v1 consists of 27 layers including 9 Inception modules. The naïve form of Inception module is restricted to filter 3 different filter sizes, which are 1×1 , 3×3 and 5×5 , while the 3×3 max pooling is performed simultaneously. The outputs are concatenated into a single output vector before it is sent to the next inception layer. However, the 5×5 convolutions filter is computationally expensive on top of a large number of filters convolutional layers. Therefore, 1×1 convolutions are added before the 3×3 and 5×5 convolutions in order to compute reductions (Szegedy et al., 2015).

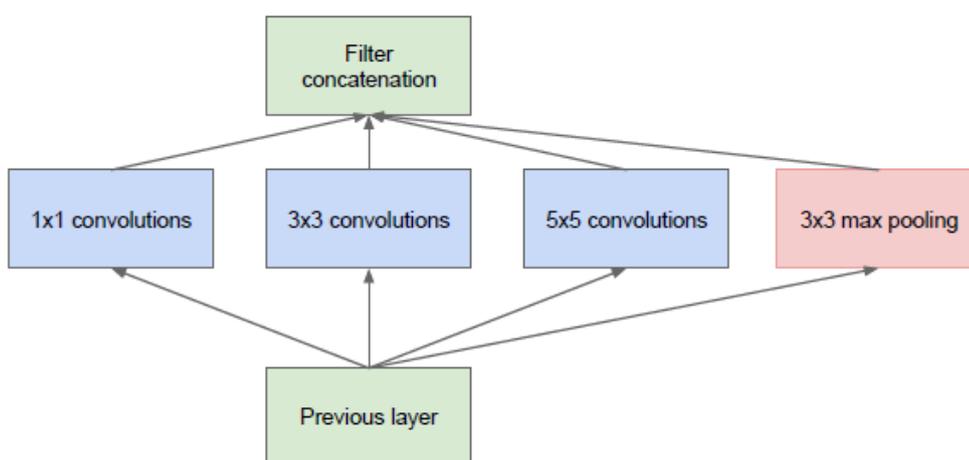


Figure 2-4: Naïve version of Inception-v1 module (Szegedy et al., 2015)

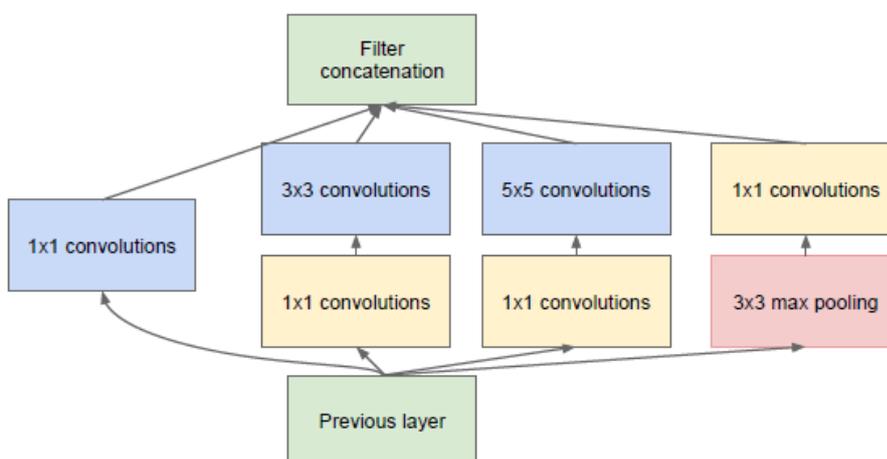


Figure 2-5: Dimension reductions inception-v1 module (Szegedy et al., 2015)

Table 1: Architecture details of Inception -v1 (Szegedy et al., 2015)

Type	Patch Size/ Stride	Output Size	Depth
convolution	7×7/2	112×112×64	1
max pool	3×3/2	56×56×64	0
convolution	3×3/1	56×56×192	2
max pool	3×3/2	28×28×192	0
inception (3a)		28×28×256	2
inception (3b)		28×28×480	2
max pool	3×3/2	14×14×480	0
inception (4a)		14×14×512	2
inception (4b)		14×14×512	2
inception (4c)		14×14×512	2
inception (4d)		14×14×528	2
inception (4e)		14×14×832	2
max pool	3×3/2	7×7×832	0
inception (5a)		7×7×832	2
inception (5b)		7×7×1024	2
avg pool	7×7/1	1×1×1024	0
dropout (40%)		1×1×1024	0
linear		1×1×1000	1
softmax		1×1×1000	0

2.4.2 Inception-v2 & Inception-v3

The second and third version of GoogleLeNet aka Inception-v2 and Inception-v3 was introduced in the same paper, “Rethinking the Inception Architecture for Computer Vision”. The performance such as the accuracy and computational complexity has improved in Inception-v2.

In Inception-v2, the 5×5 convolutions introduced in Inception-v1 has been factorized into two 3×3 convolutions to reduce the computational complexity, since a 5×5 convolution is 2.78 times computationally expensive than a 3×3 convolution (*refer to Figure 2-6*). Furthermore, the paper proposed that factorizing a n×n convolution into n×1 and 1×n could improve the computational complexity (*refer to Figure 2-7*). For example, a 3×3 convolutions filter has been factorized into two

convolutions filters, 3×1 and 1×3 to achieve a 33% cheaper computational complexity. Moreover, the filter banks outputs in Inception-v2 module were expanded to eliminate the representational bottleneck (*refer to Figure 2-8*) (Szegedy et al., 2016).

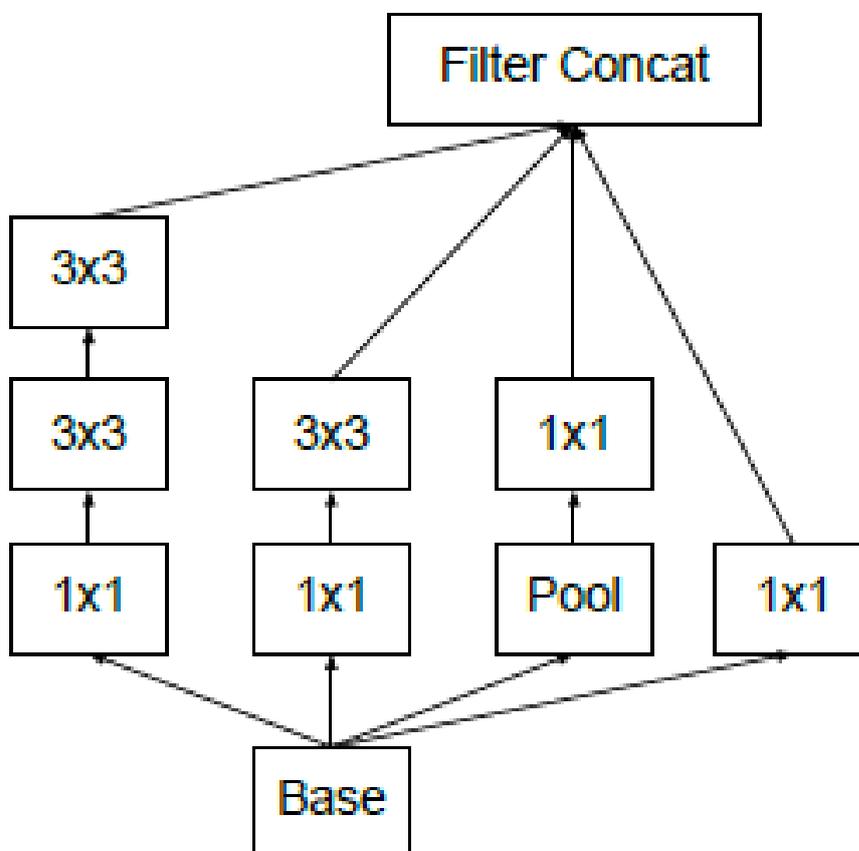


Figure 2-6: The 5×5 convolution has been replaced by two 3×3 convolution in Inception-v2 (Szegedy et al., 2016)

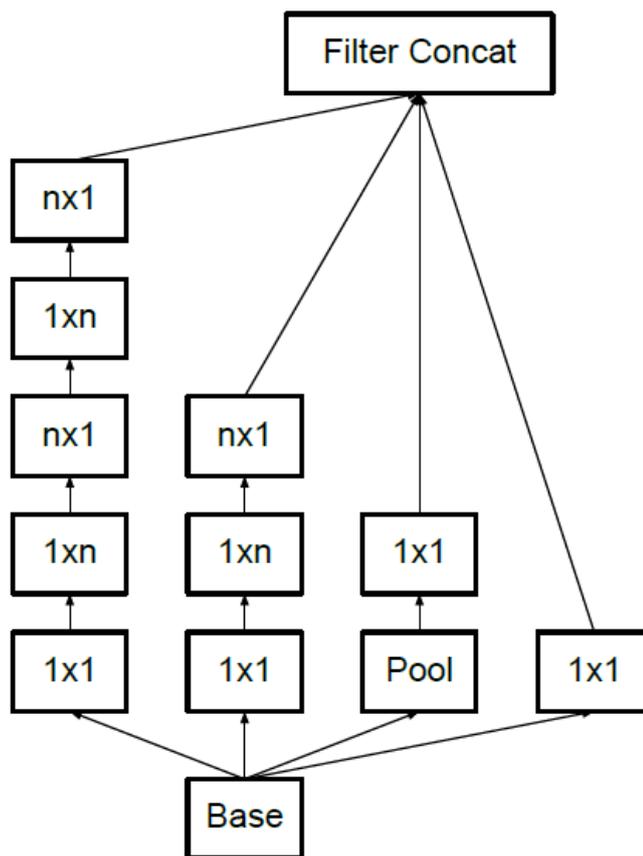


Figure 2-7: Factorization in Inception-v2 module (Szegedy et al., 2016)

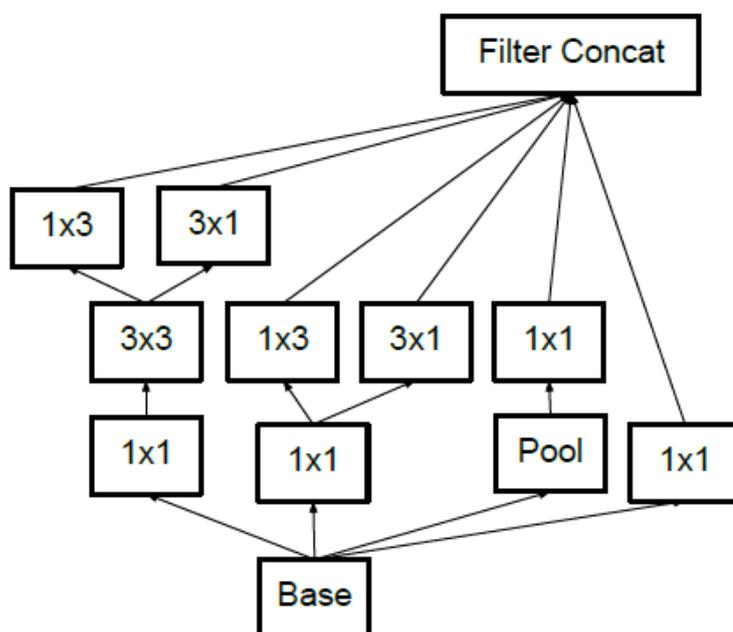


Figure 2-8: Filter banks outputs of Inception-v2 module were expanded (Szegedy et al., 2016)

Table 2: Proposed network architecture of Inception-v2 (Szegedy et al., 2016)

Type	Patch size/stride	Input size
conv	3×3/2	299×299×3
conv	3×3/1	149×149×32
conv padded	3×3/1	147×147×32
pool	3×3/2	147×147×64
conv	3×3/1	73×73×64
conv	3×3/2	71×71×80
conv	3×3/1	35×35×192
3×Inception	As in figure 5	35×35×288
5×Inception	As in figure 6	17×17×768
2×Inception	As in figure 7	8×8×1280
pool	8×8	8×8×2048
linear	logits	1×1×2048
softmax	classifier	1×1×1000

In Inception-v3, the network incorporated all improvement in Inception-v2 and the following modification:

- i) RMSProp Optimizer.
- ii) BatchNorm in Auxiliary Classifiers.
- iii) Label Smoothing.
- iv) Factorized 7 x 7 convolutions (Raj, 2018).

2.5 Residual Neural Network (ResNet)

The Residual Neural Network also known as ResNet is arguably the one of the pioneers of CNNs architecture after Inception-v1 won the ILSVRC14 with an error rate of 6.67%. ResNet achieved a top-5 error rate of 3.57% which outperform its opponent and won the 1st place in ILSVRC15.

ResNet was introduced by He et al. in the paper “Deep Residual Learning for Image Recognition” in 2015. The main idea of ResNet is to propose a deep residual learning framework to address the degradation problem occurs when deeper networks starts to converge, and accuracy becomes saturated (He et al., 2016). Hence, ResNet introduced the identity shortcut connection also called the Residual Blocks as shown in Figure 2-9. The residual blocks could skip training from multiple layers and connects to the output. The benefits of integrating the shortcut connection is to skip any layer by regularization if that particular layer could possibly affect the performance of the network.

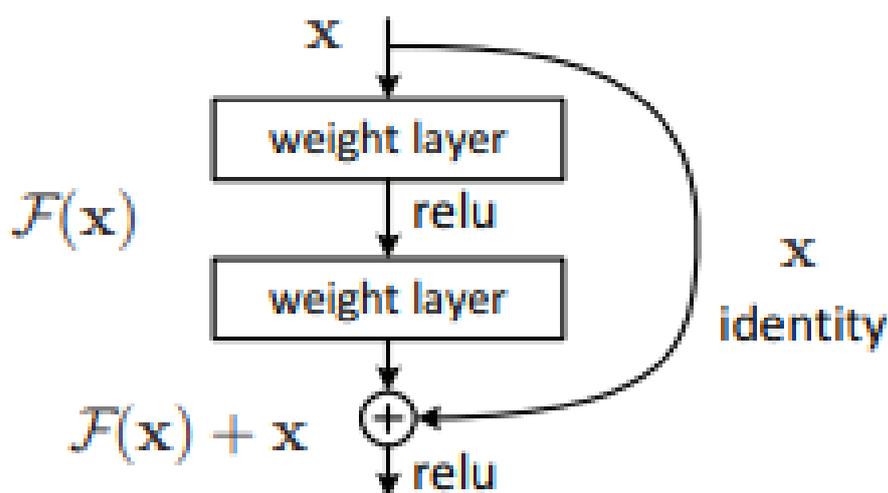


Figure 2-9: ResNet Residual Blocks (He et al., 2016)

Moreover, He et al further modified the residual blocks by introducing the pre-activation variant of the residual block as shown in Figure 2-10. In this modification, the gradients could pass through the shortcut connection to any previous layer without being interfered (Fung, 2017).

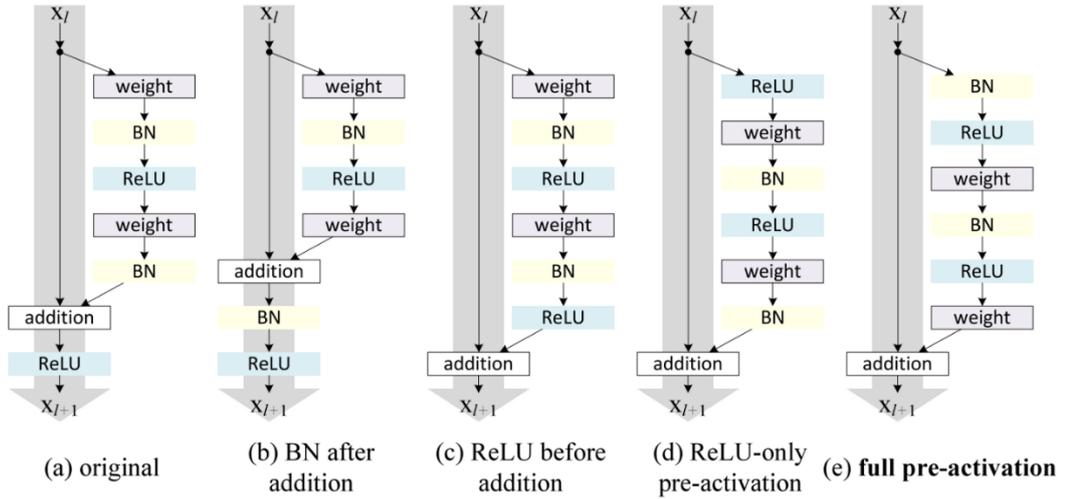


Figure 2-10: Refined residual blocks (Fung, 2017)

According to the paper “Deep Residual Learning for Image Recognition”, the shortcut connections used in ResNet was inspired by the Highway Network proposed by Srivastava et. al. in 2015. Apart from that, the similar idea of Highway Network where the information is control by the parametrized gates to flow through the shortcut connections is similar to the Long-Term Short Memory (LSTM) cell introduced by Hochreiter et. al. in 1997 cited in Fung, 2017.

Furthermore, He et al. had tested the functionality of ResNet on a plain network inspired by VGG nets and a residual network where shortcut connections are inserted. Figure 2-11 shows the network architecture comparison of the plain network and residual network. The experiment is tested on the ImageNet 2012 classification dataset consisting 1000 classes and both networks are evaluated on 18-layer and 34-layer. By referring to Table 3, ResNet has lower top-1 error compared to plain network without shortcut connections.

Table 3: Top-1 error on ImageNet Validation (He et al., 2016)

Layers	Plain network	<u>ResNet</u>
18 layers	27.94	27.88
34 layers	28.54	25.03

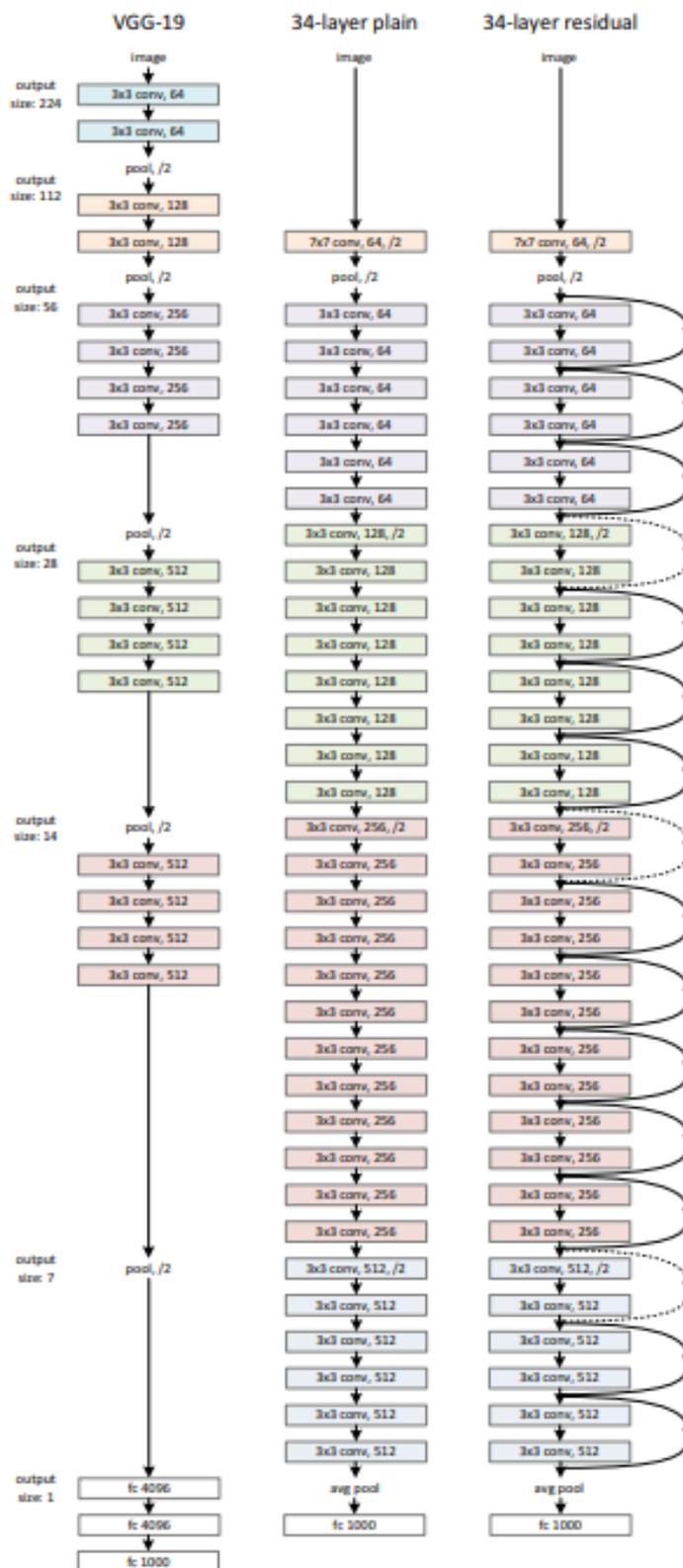


Figure 2-11: Samples of network architecture. Left: VGG-19 model. Mid: plain network inspired by VGG nets of 34 layers. Right: residual network of 34 layers (He et al., 2016)

2.6 Related Works

A set of sufficient data volume is paramount in order to train a successful deep learning model for medical image interpretation. Apart from that, skewed datasets for example 100 benign data and 1000 malignant data could leads to a bad classification results of the deep learning model. In order to overcome the low quantity and imbalance of datasets, the generative adversarial networks (GANs) is proposed to overcome the problem by generating synthetic data.

Shin et al. (2018) proposed the image-to-image translation conditional GAN (pix2pix) model introduced by Isola et al. in 2017 to produced synthetic images and classification of T1-weighted brain tumor images on Alzheimer’s Disease Neuroimaging Initiative (ADNI) datasets and Multimodal Brain Tumor Image Segmentation Benchmark (BRATS) datasets. The author has performed four approaches with different CNN input, which trained on real data only, combination of real and synthetic data, synthetic data only and synthetic data with 10% of model fine-tuning. Besides, they also implemented basic image augmentation such as rotation, crop and elastic deformation on the synthetic data. The model has achieved a mean disc-score of 0.82 which has improved accuracy than the non-augmentation GANs-based model with mean disc-score of 0.80 (CNN model trained on real and synthetic data). Shin compared their GANs-based model to the BRATS’17 best performing model, however both the GANs-based model achieves a lower accuracy.

Table 4: Disc score evaluation in terms of mean and standard deviation of GAN-based model and BRATS’17 best model (Wang et al.). The GAN-based models were trained with augmentation and without augmentation (Shin et al., 2018)

Method	Real	Real + Synthetic	Synthetic only	Synthetic only, fine-tune on 10% real
GAN-based (no aug)	0.64/0.14	0.80/0.07	0.25/0.14	0.80/0.18
GAN-based (with aug)	0.81/0.13	0.82/0.08	0.44/0.16	0.81/0.09
Wang <i>et al.</i> [20]	0.85/0.15	0.86/0.09	0.66/0.13	0.84/0.15

Iqbal & Ali (2018) introduced a new Generative Adversarial Networks model for medical imaging called MIGAN. The idea of MIGAN is for the generation of synthesis medical image. Other than the synthesis medical image, Iqbal also implement MIGAN to generate the segmented masks of the medical image. In the paper, MIGAN is applied to the retinal vessel's images for the STARE and DRIVE publicly datasets. The main contributions for this project are to generate an enhance segmented medical images then previous GANs model. Besides, MIGAN also reduce the threshold of existing GANs techniques, lesser input examples require to generate the desired synthetic images due to the refined loss function of MIGAN. Apart from that, MIGAN is less computational expensive than existing GANs model, due to the 200% improvement in the generator model during each epoch which could reduce the training period. The MIGAN structure of generator and discriminator are shown in Figure 2-13 and Figure 2-14 respectively. According to the author, the MIGAN-based deep learning model on STARE dataset has achieved disc-score of 0.838, AUC ROC of 0.985 and AUC PR of 0.922; and on DRIVE dataset has achieved disc-score of 0.832, AUC ROC of 0.984 and AUC PR of 0.916 which outperformed existing previous work.

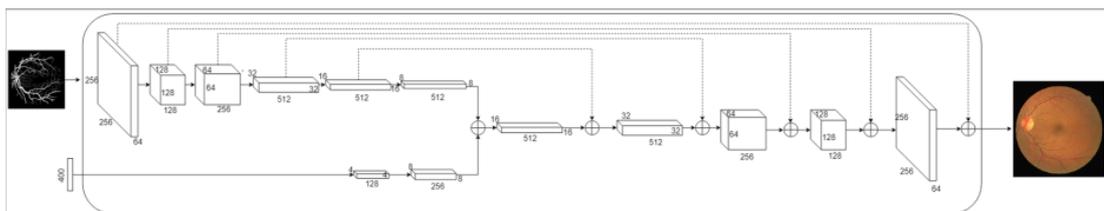


Figure 2-12: Generator structure of MIGAN (Iqbal et al., 2018)

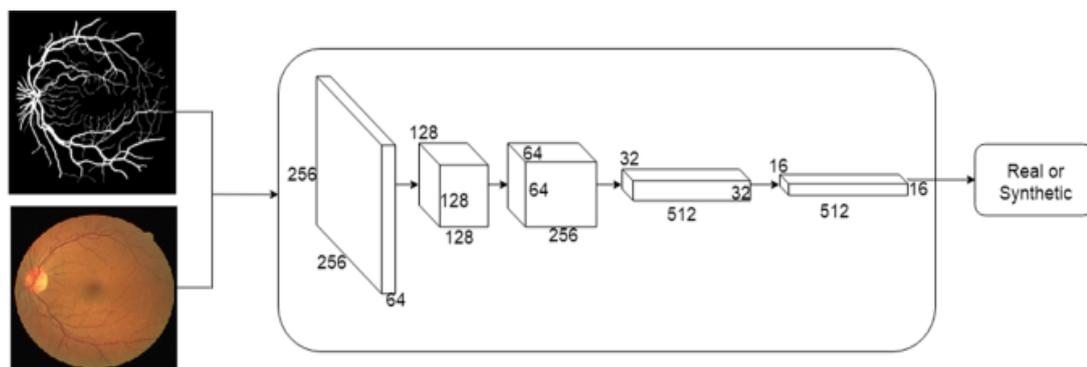


Figure 2-13: Discriminator structure of MIGAN (Iqbal et al., 2018)

Senaras et al. (2018) introduced the conditional Generative Adversarial Network also known as cGAN to generate realistic synthetic histopathological breast cancer images from the Ki67 datasets. During the preprocessing stage of the images, the operator will mark the stained nucleus manually and this stage is called the user annotation mask stage. Other than that, in the second approach the images are process though computer by using the nuclei segmentation system developed by the author in a prior study. Both the annotation mask and the nuclei segmentation obtained are feed respectively into the cGAN model as input after the preprocess stage. The generated results are analyzed by 6 researchers inclusive of 3 pathologists and 3 image analysts instead of training in a deep learning model. The average accuracy percentage of the researchers that could correctly differentiate whether the image is synthetic or real was 44.7%.

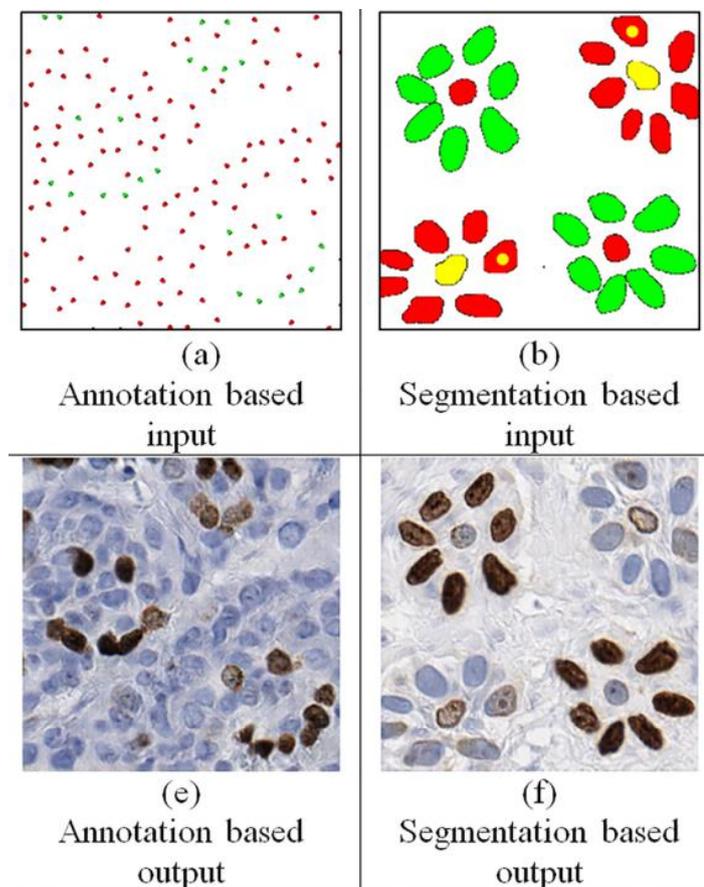


Figure 2-14: Examples of two different input and their respective generated output
(Senaras et al., 2018)

Desai et al. (2020) proposed the implementation of the Deep Convolutional GANs also known as DCGANs to generate synthetic mammogram breast cancer images from the DDSM datasets. The major objective of this work is to overcome the limited available labeled data by implementing DCGANs to generate synthetic images for deep learning breast cancer classification. The author has trained the DDSM dataset for batch size 4 and batch size 32, the samples of the synthetic images are shown in *Figure 2-16*.

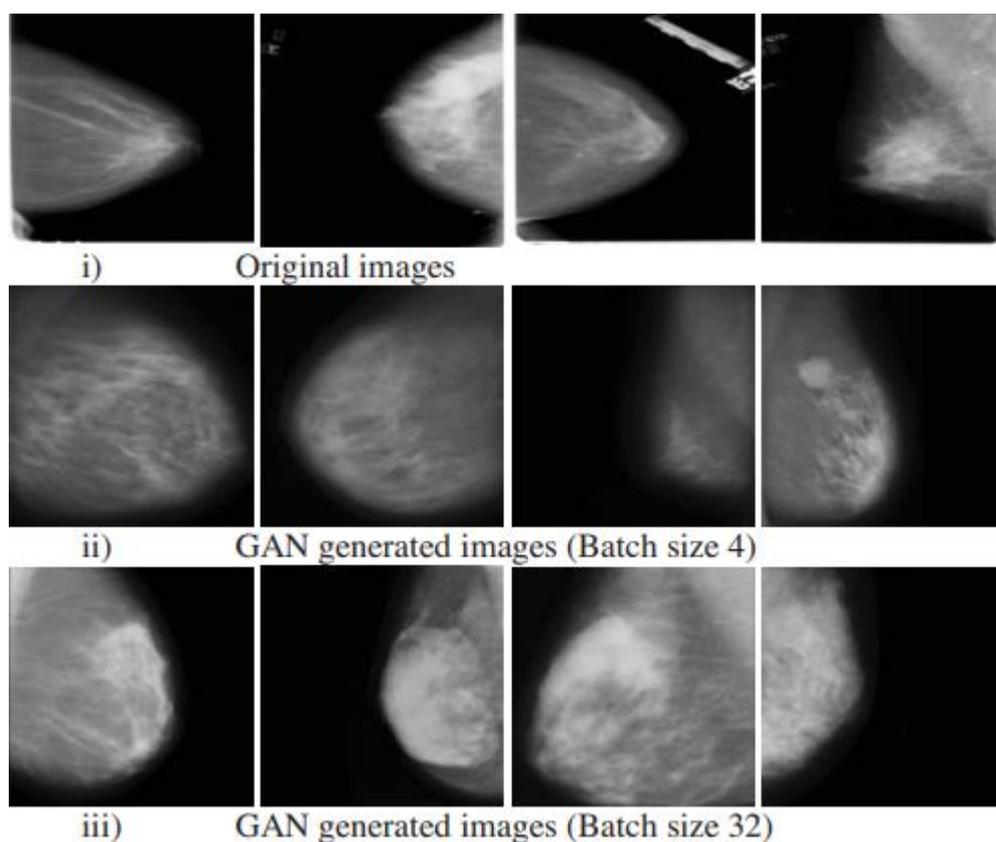


Figure 2-15: Original and GANs generated images with batch size 4 and 32 (Desai et al., 2020)

According to the CNN deep learning classification model, the batch size 32 perform better than the batch size 4 DCGANs configuration in terms of accuracy, F1-score, specificity and sensitivity. At 20 epochs, the accuracy of batch size 32 is 87%; batch size 4 is 83.58% while the accuracy without the GANs model is 78.23%. However, when the synthetic images of batch size 32 is analyzed by two professional physicians, only an average of 6 over 25 synthetic images are identified as real.

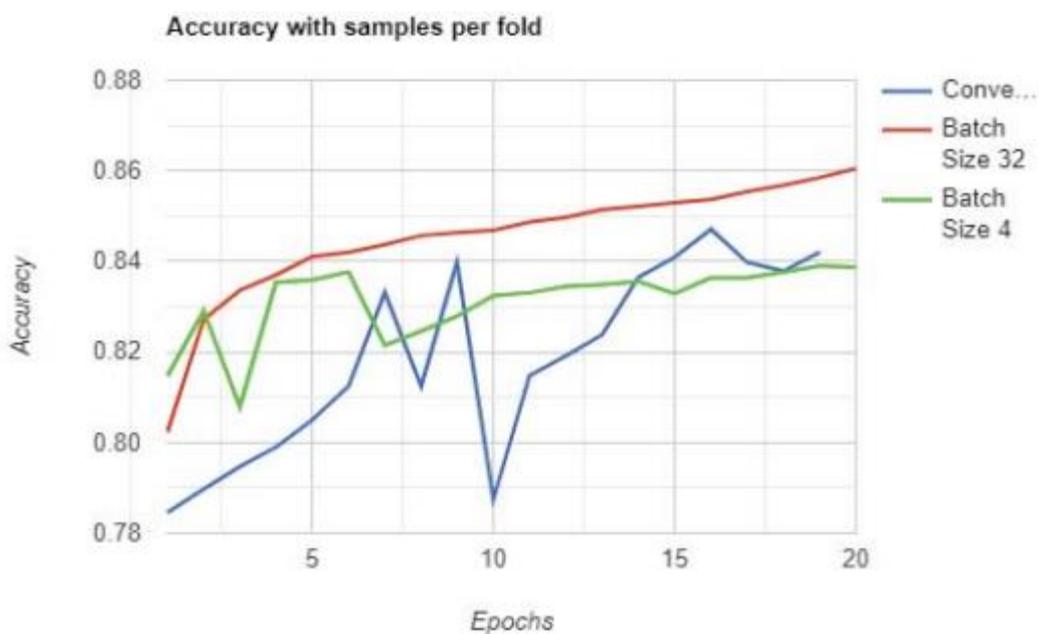


Figure 2-16: Accuracy of deep learning breast cancer classification (Desai et al., 2020)

Alyafi et al. (2020) proposed the implementation of Deep Convolutional Generative Adversarial Networks (DCGANs) to synthesized realistic and diverse mammography breast masses images to get rid of the small and imbalanced datasets obtained from OPTIMAM Mammography Image Database (OMI-DB). According to the paper, the author has trained the CNN model with four different approaches as shown in *Figure 2-17*, which are the original input (Blue), augmented original input (Orange), GANs input (Green) and augmented GANs input (Red). Furthermore, the author has augmented the both the original and synthetic data by applying random horizontal and vertical flipping. The augmentation is to improve diversity of dataset.

On the other hand, the classifier performances are recorded at $k=750$ due to the model starts to become saturated. The results shown in *Figure 2-18* illustrated that augmented GANs data as input has outperformed other inputs in terms of F1-score.

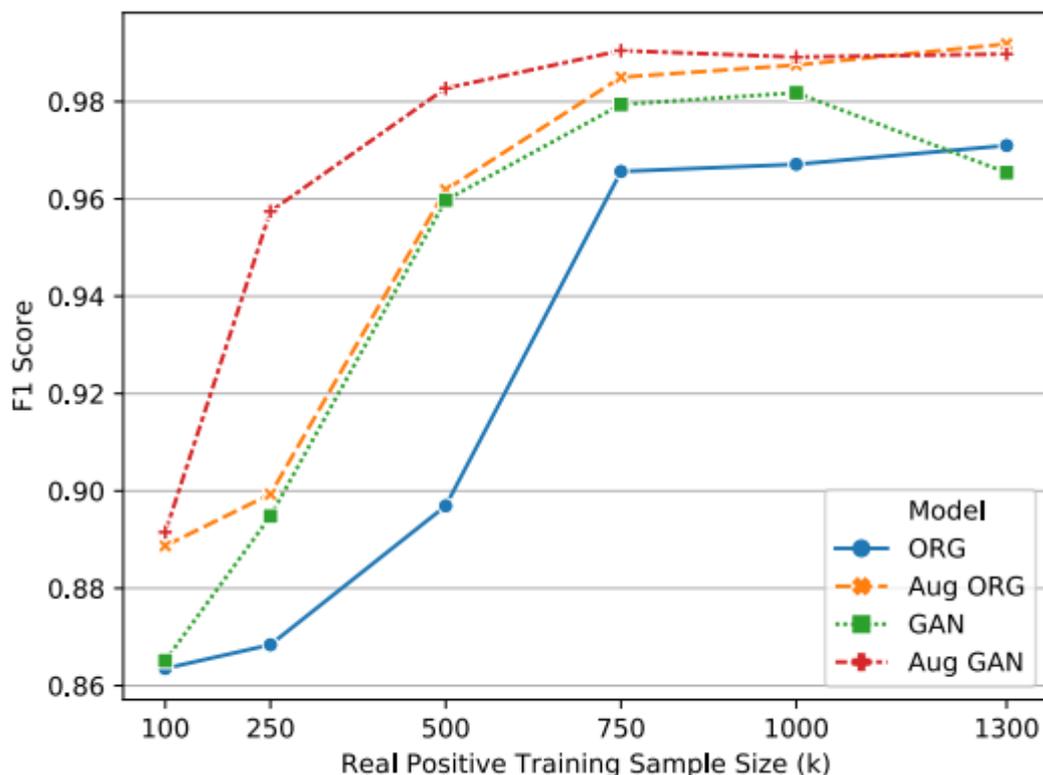


Figure 2-17: CNN predictions in terms of F1-score with four approaches: original input (Blue), augmented original input (Orange), GANs input (Green) and augmented GANs input (Red) (Alyafi et al., 2020)

Al-Dhabyani et al. (2019) proposed the implementation of Data Augmentation Generative Adversarial Networks, DAGAN and CNNs integrated with transfer learning method to classify normal, benign and malignant breast tumours. The DAGAN is inspired by the Wasserstein GAN (WGAN) introduced by Arjovsky (2017), where the resulting architecture of WGAN is used in this study. The data used for this study is the Breast Ultrasound Image (BUSI) dataset and a private dataset B. The authors have performed four approaches to train the CNNs classifier algorithm, which includes real data only; data with basic augmentation; DAGAN synthesized data; DAGAN synthesized data with basic augmentation. On the other hand, five different approaches are experimented for the CNN classifier architecture. The five architectures are CNN-AlexNet, TL-VGG16, TL-ResNet, TL-Inception and TL-NASNet. The highest accuracy achieved for this study is TL-NASNet with DAGAN synthesized data with basic augmentation at 94% (Dataset B), 92% (BUSI

Dataset) and 99% (Dataset BUSI + B). *Figure 2-19* depicted the accuracy results of the proposed dataset augmentation methods and CNNs architectures.

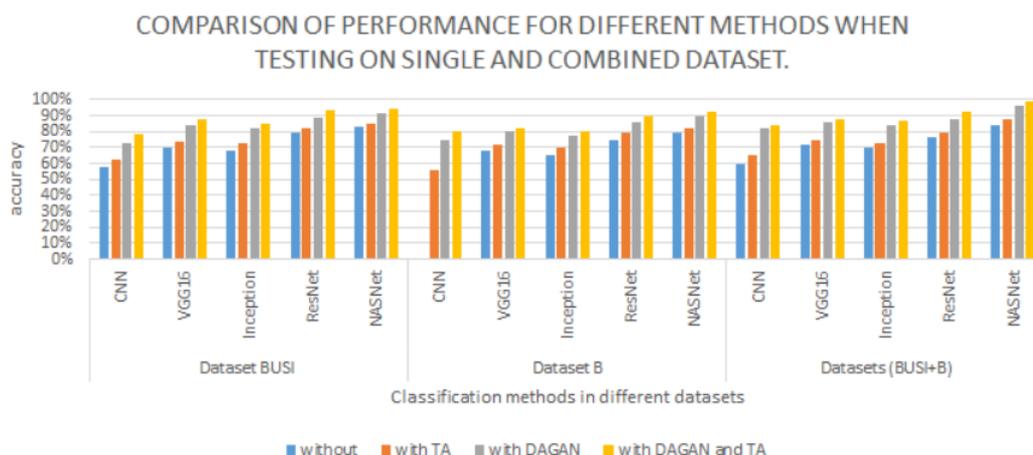


Figure 2-18: Comparison of performance in terms of accuracy on different dataset augmentation methods and CNNs architecture (Al-Dhabyani et al., 2019)

Latif et al. (2019) proposed the implementation of different deep learning and machine learning methods includes CNN model, Random Forest, Naïve Bayes, MP and SVM to classify the benign and malignant breast tumours data. The dataset used in this work is the Mendeley Breast Ultrasound (MBU) dataset. The authors have performed two approaches in processing the dataset for training, which are the unprocessed data and CNN denoised data. The highest accuracy for this achieved for this study is the CNN classifier with CNN denoise method at 88%. While the CNN classifier on the unprocessed data has the second highest accuracy at 84.02%.

Table 5: Comparison of different DL and ML classifier in terms of accuracy on unprocessed data and CNN denoised data (Latif et al., 2019)

Classifier	Unprocessed Data	CNN Denoised Data
CNN	84.02%	88.00%
Random Forest	72.97%	81.20%
Support Vector Machine	64.75%	65.73%
Multilayer Perceptron	72.34%	74.46%
Naïve Bayes	61.70%	62.00%

Khanna et al. (2021) proposed a hybrid strategy that integrated the CNN algorithm with machine learning framework to diagnose breast tumour. Breast Ultrasound Images Dataset (BUSI) was used in this work. The authors proposed that a pre-trained CNN-ResNet50 to extract features from the tumour's images, BGWO optimizer for feature selection and several SVM algorithms is used for classification. Besides, the authors also compared the performance of two approaches, which are with or without BGWO feature selection process. The highest performance in terms of accuracy is the Quadratic SVM classifier with BGWO feature selection at 84.9% and 84.6% without BGWO feature selection.

Table 6: Classification results of different machine learning classifier with or without BGWO feature selection process in terms of accuracy and AUC (Khanna et al., 2021)

Classification technique	Without feature selection		With feature selection	
	Accuracy (%)	AUC	Accuracy (%)	AUC
Linear SVM	80.9	0.96	78.8	0.95
Quadratic SVM	84.6	0.96	84.9	0.97
Cubic SVM	82.9	0.97	83.1	0.97
Medium gaussian SVM	81.8	0.97	81.4	0.97
Coarse gaussian SVM	65.9	0.94	67.3	0.93
Fine Decision tree	69.4	0.80	62.4	0.73
Medium Decision tree	68	0.87	63.1	0.79
Coarse Decision tree	64.4	0.87	61.4	0.71
Fine KNN	77.9	0.80	75.8	0.83
Medium KNN	73.2	0.91	72.1	0.91
Coarse KNN	69.6	0.89	70.6	0.89
Cosine KNN	73.2	0.92	72.2	0.91
Cubic KNN	72.9	0.91	72.9	0.91
Weighted KNN	77.4	0.94	76.9	0.94
Gaussian Naïve bayes	64.1	0.84	62.9	0.86
Kernel Naïve bayes	66.6	0.86	65.4	0.88

Gupta et al. (2022) proposed the implementation of different CNN and ML model to classify the ultrasound breast data. In this study, the Breast Ultrasound Images Dataset (BUSI) was used. Four CNN and ML classifier is implemented, which includes VGG16, VGG19, Inception-V3 and SqueezeNet integrated with KNN, SVM, RF, NN, LR and Adaboost. The highest accuracy results are the Inception-V3 model with NN algorithm at 92.6%.

Table 7: Classification results of VGG16, VGG19, Inception-V3 and SqueezeNet integrated with different ML algorithm on breast tumour classification (Gupta et al., 2022)

MODEL	VGG16				
	<i>AUC</i>	<i>Accuracy</i>	<i>F1</i>	<i>Pre</i>	<i>Recall</i>
KNN	94.2	83.3	83.1	83.4	83.3
SVM	89.8	72.5	72.8	74.9	72.5
RF	94.9	85.5	85.2	86	85.5
NN	96.8	88.5	88.4	88.4	88.5
LR	97	88.2	88.2	88.2	88.2
AdaBoost	87.1	83.8	83.6	83.8	83.8
MODEL	VGG19				
	<i>AUC</i>	<i>Accuracy</i>	<i>F1</i>	<i>Pre</i>	<i>Recall</i>
KNN	93.8	84.7	84.4	84.8	84.7
SVM	90.5	75.5	75.9	77.4	75.5
RF	94	84.5	84.2	84.8	84.5
NN	96.4	88.3	88.2	88.2	88.3
LR	97	88.6	88.5	88.6	88.6
AdaBoost	87.8	85	84.8	85	85
MODEL	Inception V3				
	<i>AUC</i>	<i>Accuracy</i>	<i>F1</i>	<i>Pre</i>	<i>Recall</i>
KNN	95.1	86.2	86.1	86.1	86
SVM	97.5	89.9	89.9	89.9	89.9
RF	94.2	83.7	83.2	84	83.7
NN	98	92.6	92.6	91.6	91.6
LR	98.1	91.1	91	91	91.1
AdaBoost	86.6	82.4	82.2	82.3	82.4
MODEL	SqueezeNet				
	<i>AUC</i>	<i>Accuracy</i>	<i>F1</i>	<i>Pre</i>	<i>Recall</i>
KNN	95.1	86.6	86.6	86.7	86.6
SVM	95.7	86.9	87	87	86.9
RF	96.4	87.6	87.4	87.7	87.6
NN	97.8	90.9	90.8	90.9	90.9
LR	97.1	89.3	89.3	89.3	89.3
AdaBoost	90.1	85.4	85.2	85.4	85.4

Table 8: Summary of performance from various related papers

GANs-based classifier				
Author	Organ	Dataset	Proposed method	Performance
Shin et al. (2018)	Brain	ADNI	pix2pix + CNN	(Mean/ Std deviation) GAN-based w/ aug: 0.82/ 0.08 GAN-based w/o aug: 0.80/ 0.07
Iqbal et al. (2018)	Eyes	STARE; DRIVE	MIGAN	(Disc-score/ AUC ROC/ AUC PR) STARE: 0.838/ 0.985/ 0.922 DRIVE: 0.832/ 0.984/ 0.916
Senaras et al. (2018)	Breast	Ki67	cGAN + analysts	44.7% by 3 pathologists and 3 image analysts.
Desai et al. (2020)	Breast	DDSM	DCGAN + CNN	(Accuracy) Batch size 4: 84% Batch size 32: 87%
Alyafi et al. (2020)	Breast	OMI-DB	DCGAN + CNN	(F1-score) GAN: 0.98 Aug GAN: 0.99
Al-Dhabyani et al. (2019)	Breast	BUSI; B	DAGAN w/ Augmentation + TL-NASNet	94% (Dataset B) 92% (BUSI Dataset) 99% (Dataset BUSI + B)
Non-GANs-based classifier				
Latif et al. (2019)	Breast	MBU	CNN	88% (CNN denoise data) 84% (Unprocessed data)
Khanna et al. (2021)	Breast	BUSI	Quadratic SVM	84.9% (with BGWO) 84.6% (without BGWO)
Gupta et al. (2022)	Breast	BUSI	Inception-V3 + NN	92.6% (Accuracy)

CHAPTER 3

METHODOLOGY

3.1 Overview

During the early stage of the project, the dataset was downloaded and preprocess. the training and validation dataset were downloaded from the Mendeley website; while the testing dataset were acquired from NCBI website. The datasets consist of two label which are the benign and malignant tumor. Subsequently, the image preprocess techniques were performed on the datasets. The image preprocessing techniques includes applying sharpening and smoothing filters on the ultrasound images to remove the image noise. Moreover, a DCGANs model is designed to generate more data to enhance the deep learning model performance by solving the problems of insufficient and unbalanced dataset. Furthermore, the synthesized data are augmented such as applying horizontal and vertical flipping and rotation to increase the variety of the dataset. The datasets were further separated into training, validation and testing set in various combinations using the k-fold cross-validation method.

After the datasets were assigned into desired combinations, the CNNs model is developed. The transfer learning model based on the literature review is implemented and fine-tuned to the datasets. Multiple CNNs architectures were tested on the datasets and the accuracy is compared and evaluated. The complete workflow is depicted as the flowchart in *Figure 3-1*. The results obtained are evaluated and compared.

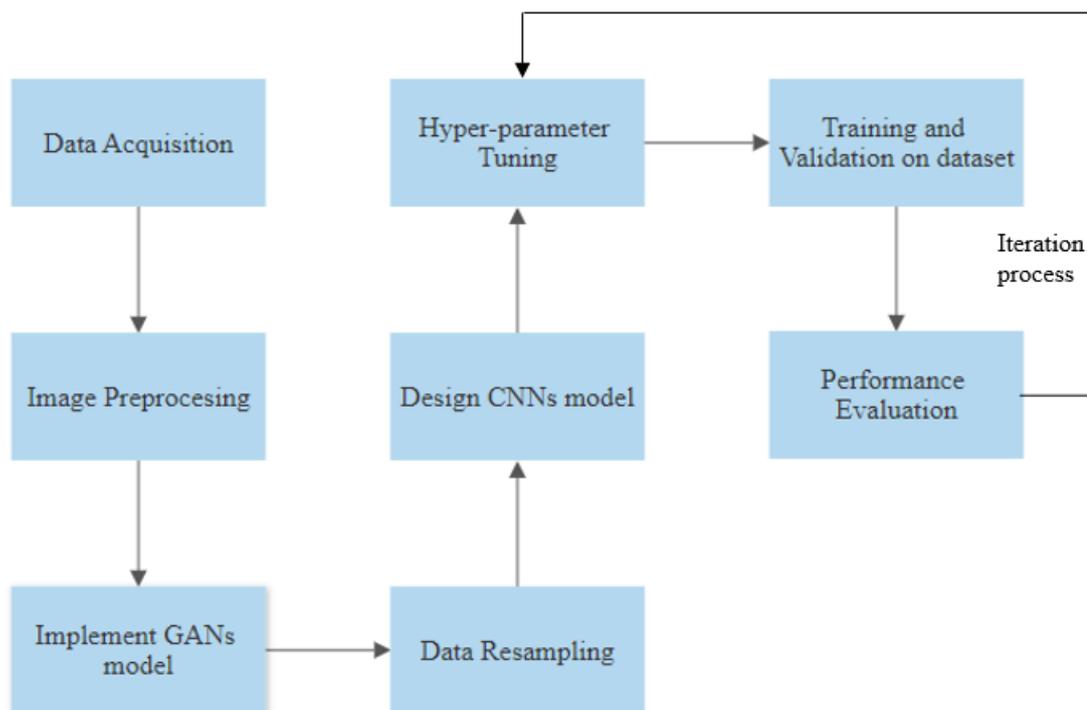


Figure 3-1: Project Methodology

3.2 Environment Setup

3.2.1 Hardware

This project is performed on an MSI Prestige 14 with an Intel i7-10510U CPU and a 2GB NVIDIA GeForce MX350 GPU equipped laptop. The hardware details are shown in *Table 9* as reference.

Table 9: Hardware details

Computer	MSI Prestige 14
CPU	Intel i7-10510U 1.80GHz
GPU	NVIDIA GeForce MX350 2GB
OS	Windows 10 Home Single Language
System type	64-bit, x64 based processor
RAM	16GB
Storage	512GB SSD

3.2.2 Software

Python 3 is used as the main programming language for this project. The DCGANs models were developed with Tensorflow, which is an open source deep learning framework developed by Facebook's AI Research lab. Besides, both the DCGANs and CNNs models were trained on Pycharm. The version of the python libraries used is shown in *Table 10* as reference.

Table 10: Python libraries version

Package	Version
Pandas	1.3.4
NumPy	1.21.3
Matplotlib	3.4.3
TensorFlow	2.9.1
Sklearn	1.0.2
OpenCV	4.5.5.64

3.3 Data Processing

3.3.1 Dataset Preparation

The dataset used for training and validation in the project is the Mendeley Breast Ultrasound dataset (MBU) by Rodrigues (2017). The dataset consists a total of 250 breast cancer ultrasound images. The dataset is separated into 100 benign tumors and 150 malignant tumors. Furthermore, the images have a low average dimension of 105×77 pixels and the file type is in BMP file. On the other hand, the testing sets of this project uses another set of Breast Ultrasound Images dataset (BUSI) by Al-Dhabyani. The average image's dimension of the BUSI images are 500×500 pixels and the file type are in PNG file. Figure 3-2 and Figure 3-3 illustrates the samples from the MBU dataset; while Figure 3-4 and Figure 3-5 illustrates the samples from the BUSI dataset.

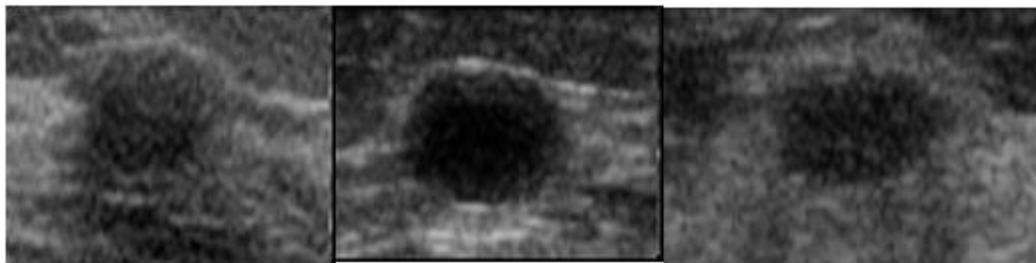


Figure 3-2: Samples of benign tumour images of MBU dataset (Rodrigues, 2017)

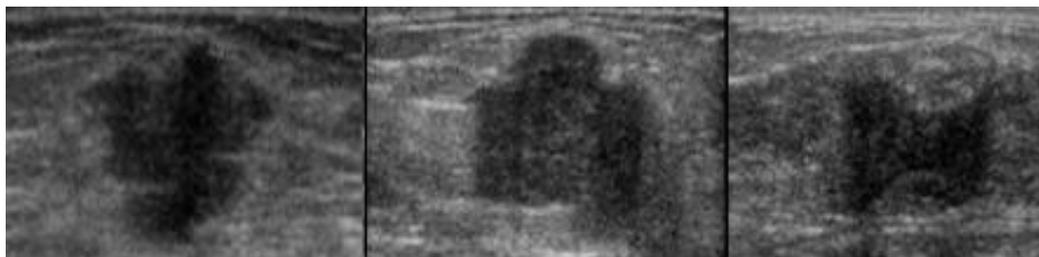


Figure 3-3: Samples of malignant tumour images of MBU dataset (Rodrigues, 2017)

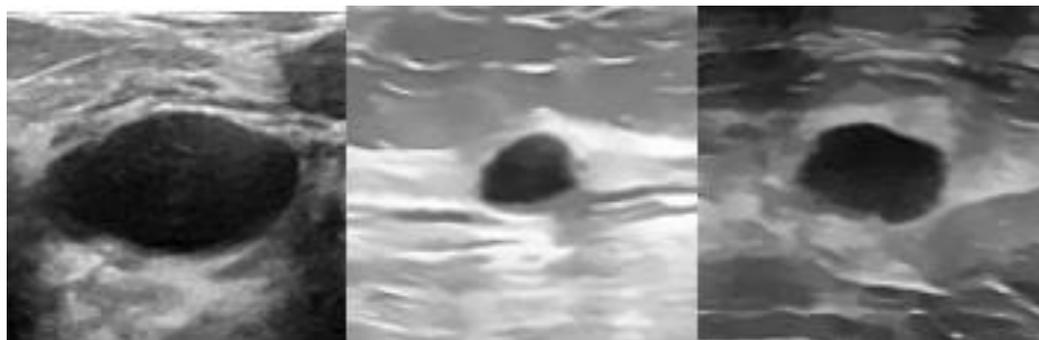


Figure 3-4: Samples of benign tumour images of BUSI dataset (Al-Dhabyani, 2020)

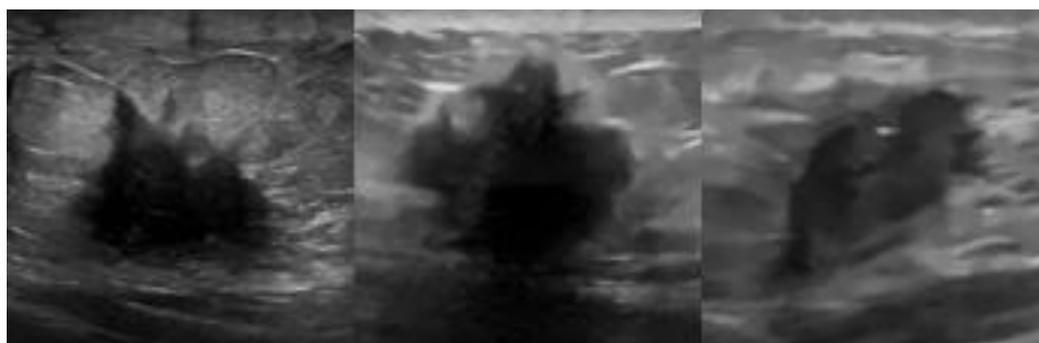


Figure 3-5: Samples of malignant tumour images of BUSI dataset (Al-Dhabyani, 2020)

3.3.2 Image Pre-Processing

After the datasets are downloaded and analyzed, some image processing methods are applied to the data in order to improve the quality of the breast tumor images. A low-pass filter kernel can be convolved to the image in order to achieve image smoothing due to its ability to remove high frequency pixel such as edges and noises in the image. Convolution can be defined as a process by applying a $n \times n$ kernel to $n \times n$ pixels over the whole image, hence transforming the image. The impact of the convolution process's transformation is determined by the size and values of the kernel, which is a matrix of values (Basavarajaiah, 2022).

By applying the 2D convolution technique, I will be utilizing my own unique kernel, therefore I have total control over the filtering procedure in this approach of smoothing. Generally, a kernel assigns a set weight to each pixel in an image and adds the weighted neighbors' pixels in order to transform that certain pixel. By implementing this approach, the pixels should be compressed in an image, reducing its clarity and making it simple to blur or smooth an image (GeeksforGeeks, 2022).

In this project, several filter and kernel are applied to the original image using OpenCV to remove the noises. In the first approach, the bilateral smoothing filter is applied to the images with a kernel size of 3×3 . In the second approach, the denoise smoothing filter is applied. In the third approach, a gaussian filter with a kernel size of 5×5 is applied to the images. In the fourth approach, a median filter with a 5×5 kernel is applied to the images. In the fifth approach, a special OpenCV filter called block matching denoise filter is applied to the image. *Table 11* below tabulate the kernel size and masks used for each filter used in the project.

Table 11: Smoothing filter implemented in this project

No	Types of filter	Kernel size	Masks																									
1	Bilateral	3x3	$\frac{1}{16} *$ <table border="1" style="display: inline-table; vertical-align: middle;"> <tr><td>1</td><td>2</td><td>1</td></tr> <tr><td>2</td><td>4</td><td>2</td></tr> <tr><td>1</td><td>2</td><td>1</td></tr> </table>	1	2	1	2	4	2	1	2	1																
1	2	1																										
2	4	2																										
1	2	1																										
2	Denoise	Null (cv2 special filter)	Null																									
3	Gaussian	5x5	$\frac{1}{273} *$ <table border="1" style="display: inline-table; vertical-align: middle;"> <tr><td>1</td><td>4</td><td>7</td><td>4</td><td>1</td></tr> <tr><td>4</td><td>16</td><td>26</td><td>16</td><td>4</td></tr> <tr><td>7</td><td>26</td><td>41</td><td>26</td><td>7</td></tr> <tr><td>4</td><td>16</td><td>26</td><td>16</td><td>4</td></tr> <tr><td>1</td><td>4</td><td>7</td><td>4</td><td>1</td></tr> </table>	1	4	7	4	1	4	16	26	16	4	7	26	41	26	7	4	16	26	16	4	1	4	7	4	1
1	4	7	4	1																								
4	16	26	16	4																								
7	26	41	26	7																								
4	16	26	16	4																								
1	4	7	4	1																								
4	Median	5x5	Median of neighbouring entries																									
5	Block Matching	Null (cv2 special filter)	Null																									

3.3.3 Image Augmentation

Due to the small dataset, several image augmentation methods have been implemented on the images. The characteristic of image augmentation of modifying current data could generate new data in different perspective for the deep learning model training process. In other words, it is the process of enhancing the dataset that is made accessible for deep learning model training.

First, the images are flipped horizontally, vertically and both horizontal and vertical, hence there will be four extra sets of data being feed into the DCGANs model. The dimension of flipped images is then resized into 64x64 pixels in the DCGANs system. The reason of resizing is to standardize the dataset. Besides, it could also improve the consistency and stability of the DCGANs and CNN network training process since the background without information is downscale and the tumours are more focused.

3.3.4 Data Augmentation

Deep Convolutional Generative Adversarial Networks also known as DCGANs is one of the most successful GANs architecture to synthesized realistic medical images. Due to the small amount of labeled dataset available for the project, instead of augmenting the images, generating realistic synthetic breast tumour images could increase the dataset amount and enhance the quality of the deep learning classifier.

DCGANs consists of a generator (G) and discriminator (D). The generator and discriminator are two different CNN model; therefore, the training process might take longer due to two deep neural network model. The DCGANs architecture was proposed by Radford et al. in 2016 and it is the modifications of the origin GANs model proposed by Goodfellow et al. in 2014 which has further improved and enhanced by many recent GANs-related papers.

Initially, the generator will input the latent dimension of noises; while the discriminator will learn from the original data and differentiate the synthetic image generated by G from the real image. The DCGANs architecture as depicted in *Figure 3-6* shows that the architecture of G and D networks are similar and inverse to each other.

In this project, the G network goes from “100x1 → 1024x4x4 → 512x8x8 → 256x16x16 → 128x32x32 → 64x64x3”, a 2D transpose is added between each layer for reshaping. A random noise vector is set to 100 as input for G and outputs a synthesized breast tumours image at size of 64x64x3. In second stage after the noise vector is input, the G network is reshaped to 1024x4x4 with a fully connected layer. Furthermore, a four fractionally-strided convolutional aka deconvolution layers with a 5x5 kernel size are added to the network. The function of the deconvolution layer is to expand the pixel by zero padded in between. Batch normalization are added to each layer except the output layer due to their characteristic to train the network independently. Besides, a LeakyReLU activation function is added to each layer except a tanh activation function is added to the output layer.

The D network is a CNN model which input an image generated by G at the image size of 64x64x3 and D will predict whether the input image is an original or synthesized image. The architecture of D is slightly straightforward compared to G, which D consists only four convolution layers with kernel size of 5x5. Like G, Batch normalization are added to each layer except the input and output layer. Additionally, a LeakyReLU activation function is added to each layer except a Sigmoid activation function is added to the output layer which produce a prediction probability between 0 and 1 (Frid-Adar et al., 2018).

The DCGANs model should trained for two different categories separately for synthesizing benign and malignant tumours. For benign, the training process was repeated for approximately 740 epochs to prevent overtraining and achieve the desired synthesized images; while for malignant, the training process was repeated for approximately 1000 epochs. The training batch size used for benign is 128, and 256 for malignant due to the amount of dataset. The DCGAN model has successfully generated 100 benign and 50 malignant images to increase and balanced the dataset. *Figure 3-6* below illustrates the increase of the dataset.

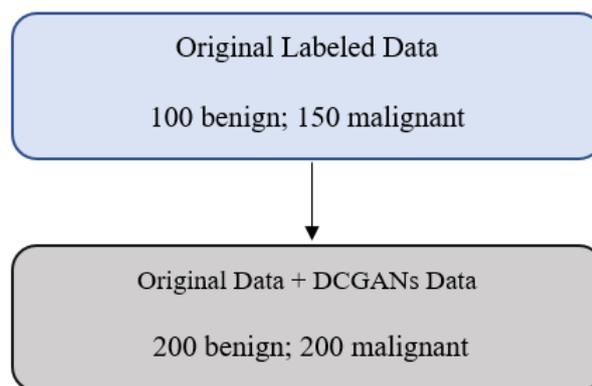


Figure 3-6: Visualization of dataset of the synthesized data and original data

3.3.5 Data Segmentation

A Train-Valid-Test split technique has been applied in order to enhance the performance of the CNN model by preventing the model to over trained from the available training data. The pre-processed datasets have been randomly split into two folders, which are training set and validation set. The training set comprised to 80% of the total dataset; while the validation data comprised to 20% of the total dataset. Synthetic images generated by the DCGANs model has been added to the dataset to increase the amount of data and balanced the dataset. Furthermore, image augmentation method such as rotating and flipping the images has been applied to the dataset to increase the amount of data. Hence, the total dataset has been increased to 20,800 images which consists of 10,400 benign data and 10,400 malignant data.

By implementing the Train-Valid split technique, 8,320 benign images and 8,320 malignant images have been randomly assigned to the training dataset (80% of total dataset); The validation dataset consists of 2,080 images for each category (20% of total dataset). Since the datasets are split randomly, therefore the results obtained are strictly not biased. *Figure 3-7* depicted the overview of data segmentation in this project. After the model has trained successfully, the model is tested with the BUSI dataset which is a completely different source from the training dataset. BUSI dataset consists of 480 images which includes 240 benign images and 240 malignant images.

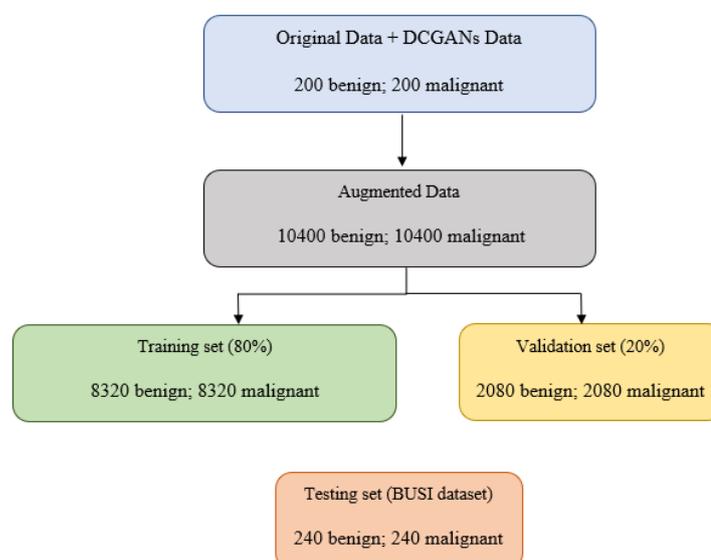


Figure 3-7: Visualization of dataset distribution

3.4 Classification Model

3.4.1 Dataset Cross-Validation

Apart from separating the datasets into training, validation and testing dataset, the cross-validation method could shuffle the datasets in a more thorough way. The accuracy of the CNN model may be saturated easily after a short amount of iterations with the same training and validation set. In this case, the CNN model's score such as accuracy, precision and F1-score may seem to have a perfect result, but it would fail to predict any random unseen data, therefore the network is considered as over-trained in this circumstance.

In this project, the k-fold cross validation is implemented to split and shuffle the training and validation dataset. This method can be applied in order to tune the hyperparameters so that the model could be trained with the best hyperparameter value. The benefit of this method is that every training and validation set can be utilised for one time, therefore the model would not saturate easily and have a fairer validating process. Hence, the CNN model can train and validate on k number of different datasets, to ensure the model is more generalized (Kumar, 2022).

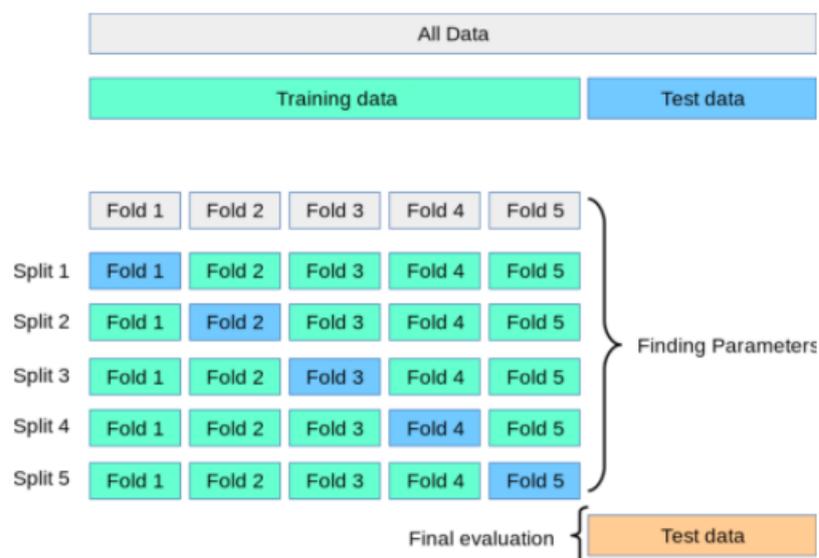


Figure 3-8: Terminology of 5-fold cross validation (Kumar, 2022)

A stratified 5-fold cross validation is applied to the CNN model in this project. *Figure 3-8* above illustrates the mathematical concept and the training and test set distribution of k-fold cross validation, $k = 5$. The ultrasound breast tumour images are partitioned into 5 different combinations of training and testing set. The CNN model will undergo 5 iterations of training and validation process. Lastly, all 5 results generated are average divided to obtain the generalized estimation score.

3.4.2 CNNs and TL Architecture Design

3.4.2.1 CNNs-AlexNet

In this project, the AlexNet architecture is proposed to implement in this CNN breast tumour classifier. The AlexNet architecture was introduced by Alex Krizhevsky in 2012. The architecture of AlexNet consists of three fully connected layers and five convolutional layers integrated with three max pooling layers. *Figure 3-9* below illustrates the architecture of AlexNet. AlexNet uses a ReLU activation function at the end of each layer except for the last layer. The advantages of performing the ReLU activation function is due to its speedy training time compared to the tanh activation. The last layer of the AlexNet architecture outputs with a softmax function due to the 2-labelled classification of benign and malignant tumours. Furthermore, dropout function is implemented in the first two fully connected layers. The dropout function could turn off the neurons with a specific probability in order to avoid overfitting.

Table 12: Configuration used in CNN-AlexNet

Layer	Type	Maps	Size	Kernel Size	Padding	Activation Function
In	Input	64	100*100	-	-	-
C1	Convolution	64	98*98	3*3	Same	eLU
S2	Max Pooling	64	49*49	2*2	Valid	-
C3	Convolution	32	49*49	3*3	Same	eLU

C4	Convolution	32	47*47	3*3	Valid	eLU
S5	Max Pooling	32	23*23	2*2	Valid	-
C6	Convolution	16	23*23	3*3	Same	eLU
C7	Convolution	16	21*21	3*3	Valid	eLU
S8	Max Pooling	16	10*10	2*2	Valid	-
F9	Fully connected	-	64	-	-	eLU
F10	Fully connected	-	32	-	-	eLU
F11	Fully connected	-	16	-	-	eLU
Out	Fully connected	-	2	-	-	Softmax

Table 8 above tabulates the general configuration of the proposed AlexNet architecture used in the CNN classifier. The model consists of five 2D convolutional layers and three pooling layers which is a typical structure of an AlexNet architecture. The dropout is set as 0.25 after each of the pooling layers. Besides, the activation function used in each 2D convolutional layer is the eLU. The eLU activation function is used due to its ability to smooth slowly through the output reaches $-\alpha$, while the ReLU would smooth sharply, therefore eLU tends to perform better in terms of accuracy. Lastly, one flattens layer and three dense layers are used to transform the matrix into a single array to allow the softmax function to generate results accurately. The CNNs-AlexNet algorithm uses the Adam optimizer with a default learning rate of 0.001. Besides, the algorithm uses binary cross entropy loss function to evaluate the performance of the model.

Layer (type)	Output Shape	Param #
conv2d (Conv2D)	(None, 100, 100, 64)	1792
conv2d_1 (Conv2D)	(None, 98, 98, 64)	36928
max_pooling2d (MaxPooling2D)	(None, 49, 49, 64)	0
dropout (Dropout)	(None, 49, 49, 64)	0
conv2d_2 (Conv2D)	(None, 49, 49, 32)	18464
conv2d_3 (Conv2D)	(None, 47, 47, 32)	9248
max_pooling2d_1 (MaxPooling2D)	(None, 23, 23, 32)	0
dropout_1 (Dropout)	(None, 23, 23, 32)	0
conv2d_4 (Conv2D)	(None, 23, 23, 16)	4624
conv2d_5 (Conv2D)	(None, 21, 21, 16)	2320
max_pooling2d_2 (MaxPooling2D)	(None, 10, 10, 16)	0
dropout_2 (Dropout)	(None, 10, 10, 16)	0
flatten (Flatten)	(None, 1600)	0
dense (Dense)	(None, 64)	102464
dropout_3 (Dropout)	(None, 64)	0
dense_1 (Dense)	(None, 32)	2080
dropout_4 (Dropout)	(None, 32)	0
dense_2 (Dense)	(None, 16)	528
dropout_5 (Dropout)	(None, 16)	0
dense_3 (Dense)	(None, 2)	34
=====		
Total params: 178,482		
Trainable params: 178,482		
Non-trainable params: 0		
=====		

Figure 3-9: Model summary of the CNN-AlexNet architecture implemented

3.4.2.2 TL-Inception-V3 with 3 extra hidden layers + dropout

The transfer learning technique can be applied to the project in order to utilise the improved version of ILSVRC14 winner, Inception-V3 as a pre-trained model in this breast tumour classifier project. The weights of Inception-V3 model were trained on a gigantic amount of dataset using several high-powered GPUs, and the transfer learning technique allows the model to be implemented in the classifier of this project (Irla, 2019). Besides, due to the small dataset available to this project, hence the proposed pre-trained Inception-V3 model could be beneficial to the classifier.

All layers of the Inception-V3 except for the last fully connected is imported to the classifier of the project. Besides, all the layers are set to non-trainable and some lower layers are added, therefore the classifier could train the tumours data on the lower layers while keeping the trained-parameters of Inception-V3 constant. Four extra layers were added to the TL-InceptionV3 model, including 1 average pooling layer with 0.2 dropout, 1 flatten layer, and 1 fully connected 128-size layer. The output layer uses the softmax activation function to classify the benign and malignant tumours class. Moreover, the binary cross entropy is used as the algorithm loss function due to the two target classes of output. Besides, the Adam optimizers with a default learning rate of 0.001 is used at the output layer. *Figure 3-10* below illustrates the general overview of the proposed TL-InceptionV3 architecture.

Layer (type)	Output Shape	Param #
inception_v3 (Functional)	(None, 1, 1, 2048)	21802784
global_average_pooling2d (GlobalAveragePooling2D)	(None, 2048)	0
dropout (Dropout)	(None, 2048)	0
flatten (Flatten)	(None, 2048)	0
dense (Dense)	(None, 128)	262272
dense_1 (Dense)	(None, 2)	258
Total params: 22,065,314 Trainable params: 262,530 Non-trainable params: 21,802,784		

Figure 3-10: Model summary of the TL-Inception-V3 architecture implemented

3.4.2.3 TL-DenseNet with 6 extra hidden layers + dropout

DenseNet is chosen for the second transfer learning model due to the simplicity in its algorithm architecture. Since most of the CNNs architecture are getting deeper, thus the information from the input layer could be faded away before arriving the output layer. Besides, DenseNet also required lesser parameters, hence it could decrease the training time. Furthermore, according to the literature review, the implementation of TL-DenseNet on the BUSI and MBU datasets is still unprecedented by previous works, therefore it may be a great opportunity for contributing to related studies.

Similar to TL-Inception-V3, all layers except the last fully connected layer of TL-DenseNet are imported to the CNN classifier. All layers are set as non-trainable, and 6 extra hidden layers are added to the architecture, which includes 1 average pooling layer, 2 batch normalization layers with dropout of 0.5, and 3 fully connected layers. The extra hidden layers are trained to classify the benign and malignant tumours data. The output layer uses the softmax activation function to classify the benign and malignant tumours class. Moreover, the categorical cross entropy is used as the algorithm loss function. Besides, the Adam optimizers with a default learning rate of 0.001 is used at the output layer. *Figure 3-11* below illustrates the general overview of the proposed TL-DenseNet architecture.

Layer (type)	Output Shape	Param #
densenet121 (Functional)	(None, 3, 3, 1024)	7037504
global_average_pooling2d (GlobalAveragePooling2D)	(None, 1024)	0
batch_normalization_94 (BatchNormalization)	(None, 1024)	4096
dropout (Dropout)	(None, 1024)	0
dense (Dense)	(None, 1024)	1049600
dense_1 (Dense)	(None, 512)	524800
batch_normalization_95 (BatchNormalization)	(None, 512)	2048
dropout_1 (Dropout)	(None, 512)	0
dense_2 (Dense)	(None, 2)	1026
=====		
Total params: 8,619,074		
Trainable params: 1,578,498		
Non-trainable params: 7,040,576		
=====		

Figure 3-11: Model summary of the TL-DenseNet architecture implemented

3.5 Evaluation Method

The evaluation metrics used for the training and validation process are loss and accuracy. The accuracy is defined as the amount of correct predictions. The performance of the algorithm is evaluated using an accuracy metric. A model's accuracy is often assessed after the model's input parameters and is expressed as a percentage. It measures how closely your model's forecast matches the actual data. For example, for 100 test samples, if the classifier successfully predicts 95 samples correctly, thus the classifier's accuracy will be 95%. On the other hand, the loss metrics is defined as the difference between the predicted and true value of the model. Generally, the loss function is used to optimize a deep learning model by comparing the performance of the model on training and validation set after each iteration.

For the testing set evaluation metrics, the F1-score, accuracy, precision and the confusion matrix are implemented. *Table 13* below tabulate the terminology of the confusion matrix. In order to enhance the performance of the deep learning model, the model should increase the TP and TN predictions and minimize the FP and FN predictions. Furthermore, precision is defined as the percentage of true positive over the total predicted positive amount; while the F1-score is defined as the harmonic mean of combination between precision and recall.

Table 13: Terminology of Confusion Matrix

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

The calculations of the evaluation metrics are shown below:

$$\text{Accuracy} = (\text{TP} + \text{TN}) / \text{All Predictions}$$

$$\text{Precision} = \text{TP} / \text{Predictions Positive}$$

$$\text{Recall} = \text{TP} / (\text{TP} + \text{FN})$$

$$\text{F1 Score} = 2 * (\text{Recall} * \text{Precision}) / (\text{Recall} + \text{Precision})$$

3.6 Project Timeline

Research on GANs and deep learning model has been performed in the first two weeks of the first trimester before confirming the FYP topic. After confirming the FYP topic with my supervisor, Dr. Humaira Nisar, the data is received and downloaded from the Mendeley website. Besides, the testing dataset is acquired from the NCBI website. The downloaded datasets were analysed and evaluate. The dataset images were pre-processed by implementing image process method such as image smoothing, flipping and rotating. The data acquisition and pre-process stage took approximately two weeks.

In the second phase, the DCGANs model was studied and designed accordingly to the research paper “*Unsupervised Representation Learning with Deep Convolutional Generative Adversarial Network*” proposed by Radford et al. in 2015. The training process took more than 2 months to synthesized desired benign tumors images. The training process has been repeated multiple times in order to obtain desired results and the computational power of the GANs model is very high, therefore the training takes longer period. With existing training experience of benign data, the malignant data training process took a shorter period to generate desired results.

After the all desired synthesized image has been generated, the synthesized images were added into the original dataset to increase the amount of dataset and to balance the benign and malignant dataset. The synthesized tumors images were preprocessed to remove the image noises. Furthermore, data augmentation for instance flipping, rotation is performed on the synthesized images. The complete datasets consisting 15,600 data were resampling and separated randomly into training, validation and testing set. The data augmentation and resampling took approximately one week.

In the third phase, several CNN models such as Inception-V3, ResNet50, AlexNet and DenseNet are designed and trained on the augmented dataset. The deep learning model and fine-tuning process took approximately five weeks. The final performance evaluation is completed in Week 12 of the second trimester.

CHAPTER 4

RESULTS AND DISCUSSIONS

4.1 Overview

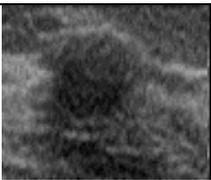
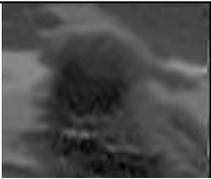
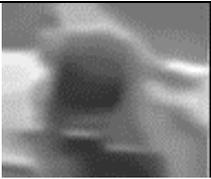
The major objective of this project is to develop a deep learning neural network classifier algorithm that could distinguish the benign and malignant breast tumours in any form of two-dimensional picture. The Mendeley Breast Ultrasound dataset (MBU) by Rodrigues (2017) was acquired from the Mendeley website for the CNNs algorithm training and validation purpose. The original 250 data inclusive of benign and malignant images were increased to 20,800 images by adding the synthesized images generated by DCGANs and image augmentation techniques were implemented to the images. Besides, smoothing filters were applied to the dataset to reduce the noises of the ultrasound images in order to enhance the image quality.

The data were further randomly split into training, validation set with a percentage of 0.8, 0.2. Besides, a 5-fold cross validation is implemented to ensure the training and validation data were shuffled. Besides, the testing dataset, BUSI was acquired from a different source to evaluate the classifier. Several CNNs model such as Inception-V3, AlexNet and DenseNet were developed and trained. The results obtained were recorded and compared. On the other hand, another set of open-source breast ultrasound image, BUSI dataset (Al-Dhabyani et al., 2020) acquired from NCBI is used as the testing set of the CNNs algorithm. The performance of the deep learning classifier model is evaluated using confusion matrix, accuracy, precision, recall rate and F1-score. The results for Inception-V3, AlexNet and DenseNet are compared and discussed.

4.2 Image Pre-Processing

Five different smoothing filters such as bilateral, denoise, gaussian, median and block matching filter have been applied on the original ultrasound image. According to the filtered images, the Denoise filter seems to be the most suitable filter to remove the ultrasound image's noises and still maintain the edge of the tumours, therefore the Denoise filter is chosen for the project and it is applied to all images in the dataset. *Table 15* below illustrates the pre-processed images of different filters.

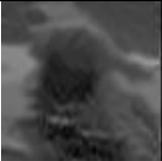
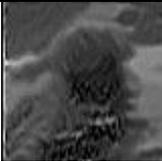
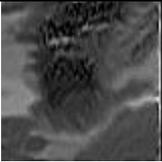
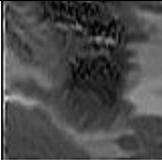
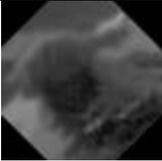
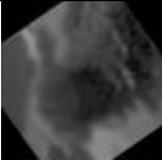
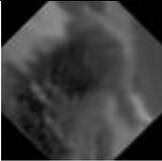
Table 15: Pre-processed filtered images

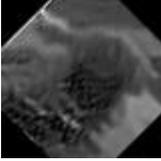
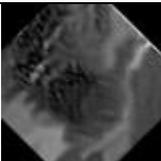
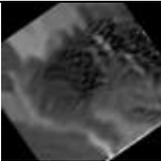
	Filter	Pre-processed image	Masks																									
1	None (Original)		Null																									
2	Bilateral		$\frac{1}{16} *$ <table border="1" data-bbox="1214 992 1370 1149"> <tr><td>1</td><td>2</td><td>1</td></tr> <tr><td>2</td><td>4</td><td>2</td></tr> <tr><td>1</td><td>2</td><td>1</td></tr> </table>	1	2	1	2	4	2	1	2	1																
1	2	1																										
2	4	2																										
1	2	1																										
3	Denoise		Null																									
4	Gaussian		$\frac{1}{273} *$ <table border="1" data-bbox="1185 1382 1393 1579"> <tr><td>1</td><td>4</td><td>7</td><td>4</td><td>1</td></tr> <tr><td>4</td><td>16</td><td>26</td><td>16</td><td>4</td></tr> <tr><td>7</td><td>26</td><td>41</td><td>26</td><td>7</td></tr> <tr><td>4</td><td>16</td><td>26</td><td>16</td><td>4</td></tr> <tr><td>1</td><td>4</td><td>7</td><td>4</td><td>1</td></tr> </table>	1	4	7	4	1	4	16	26	16	4	7	26	41	26	7	4	16	26	16	4	1	4	7	4	1
1	4	7	4	1																								
4	16	26	16	4																								
7	26	41	26	7																								
4	16	26	16	4																								
1	4	7	4	1																								
5	Median		Median of neighbouring 5x5 entries																									
6	Block Matching		Null																									

4.3 Image Augmentation

Nine different image augmentation techniques are implemented on the pre-processed images. The dimension of the pre-processed images was resized to 100*100 pixels before the augmentation process. The dataset is increased in size after the augmentation process. *Table 16* below illustrates the samples of the augmented benign tumour images.

Table 16: Samples of augmented image

	Image Orientation	Augmented Image
1	Upright	
2	Horizontal flipping	
3	Vertical flipping	
4	Horizontal Vertical flipping	
5	Anticlockwise 45° rotation	
6	Anticlockwise 125° rotation	
7	Anticlockwise 315° rotation	

8	Rotation horizontal flip	
9	Rotation vertical flip	
10	Rotation horizontal vertical flip	

4.4 Data Augmentation using DCGAN

The training process of the DCGANs model for benign data took 800 epochs and the training duration is approximate 28 hours. While the training process of the DCGANs model for malignant data took 1000 epochs and the training duration is approximate 36 hours. However, the training process is repeated several iterations to generate the most realistic synthesized image. *Table 17* below illustrates the samples of the synthesized image generated by the DCGANs model. Furthermore, the synthesized images were pre-processed by applying the Denoise filter and augmented using the techniques mentioned above.

Table 17: Samples of synthesized image generated by DCGANs

	Tumours type	Synthesized image
1	Benign	
2	Benign	

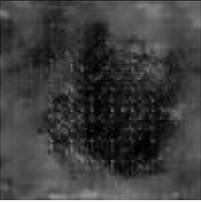
3	Malignant	
4	Malignant	

Table 18 below tabulate the number of the augmented images in the dataset. The dataset has been increased to 20800 data which consists of 10400 benign data and 10400 malignant data.

Table 18: Number of images in the dataset

	Image Orientation	Number of images
1	Upright	1-250
2	DCGAN synthesized image	251-400
2	Horizontal flipping	401-800
3	Vertical flipping	801-1200
4	Horizontal Vertical flipping	1201-1600
5	Anticlockwise 45° rotation	1601-3200
6	Anticlockwise 125° rotation	3201-4800
7	Anticlockwise 315° rotation	4801-6400
8	Rotation horizontal flip	6401-11200
9	Rotation vertical flip	11201-16000
10	Rotation horizontal vertical flip	16001-20800

4.5 Training Results

According to Table 14 below, all three proposed models, CNN-AlexNet model, TL-Inception-V3 with 3 extra hidden layers + dropout model and TL-DenseNet with 6 extra hidden layers + dropout model successfully produced all evaluation metrics, accuracy, precision, recall and F1-score rates above 90%. As illustrated in *Figure 4-1*, the bar chart illustrates the comparison of the proposed models on the validation dataset, the TL-DenseNet model performs best in terms of accuracy at 97.61%; while the CNN-AlexNet achieved the highest F1-score among all proposed models at 0.9950.

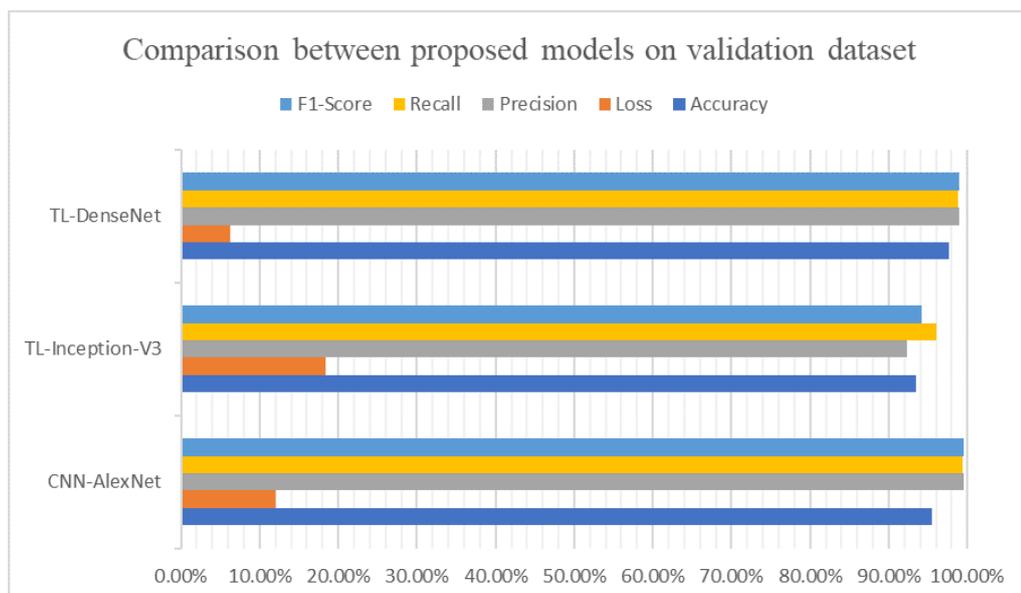


Figure 4-1: Bar chart of comparison between proposed model on validation dataset

Table 19: Comparison between proposed model in terms of accuracy, loss, precision, recall and F1-score on the validation dataset

Model	Accuracy	Loss	Precision	Recall	F1-Score
CNN-AlexNet	95.52%	0.1210	0.9958	0.9942	0.9950
TL-Inception-V3	93.46%	0.1833	0.9238	0.9605	0.9413
TL-DenseNet	97.61%	0.0625	0.9896	0.9883	0.9889

4.5.1 CNN-AlexNet

The training process took 10 epochs and 5-fold cross validation. The average training duration of one epoch is approximately 10 minutes and the total training time for 10 epochs and 5-fold cross validation process is approximately 8.3 hours. The CNNs-AlexNet architecture achieved an average validation accuracy of 95.52%; average validation loss of 0.1210; average validation precision of 0.9958; average validation recalls of 0.9942; average validation F1-Score of 0.9950.

The complete 5-fold training results of CNN-AlexNet model are attached in the Appendix A section below.

Table 20: Validation Evaluation Metrics for CNN-AlexNet in Each Fold

No of Folds	Accuracy	Loss	Precision	Recall	F1-Score
1	80.90%	0.3778	0.9814	0.9736	0.9775
2	99.23%	0.0174	1.0000	0.9984	0.9992
3	98.66%	0.0395	0.9992	1.0000	0.9996
4	99.97%	0.0016	1.0000	0.9992	0.9996
5	98.86%	0.0119	0.9984	1.0000	0.9992
Average	95.52%	0.1210	0.9958	0.9942	0.9950

4.5.1.1 Accuracy

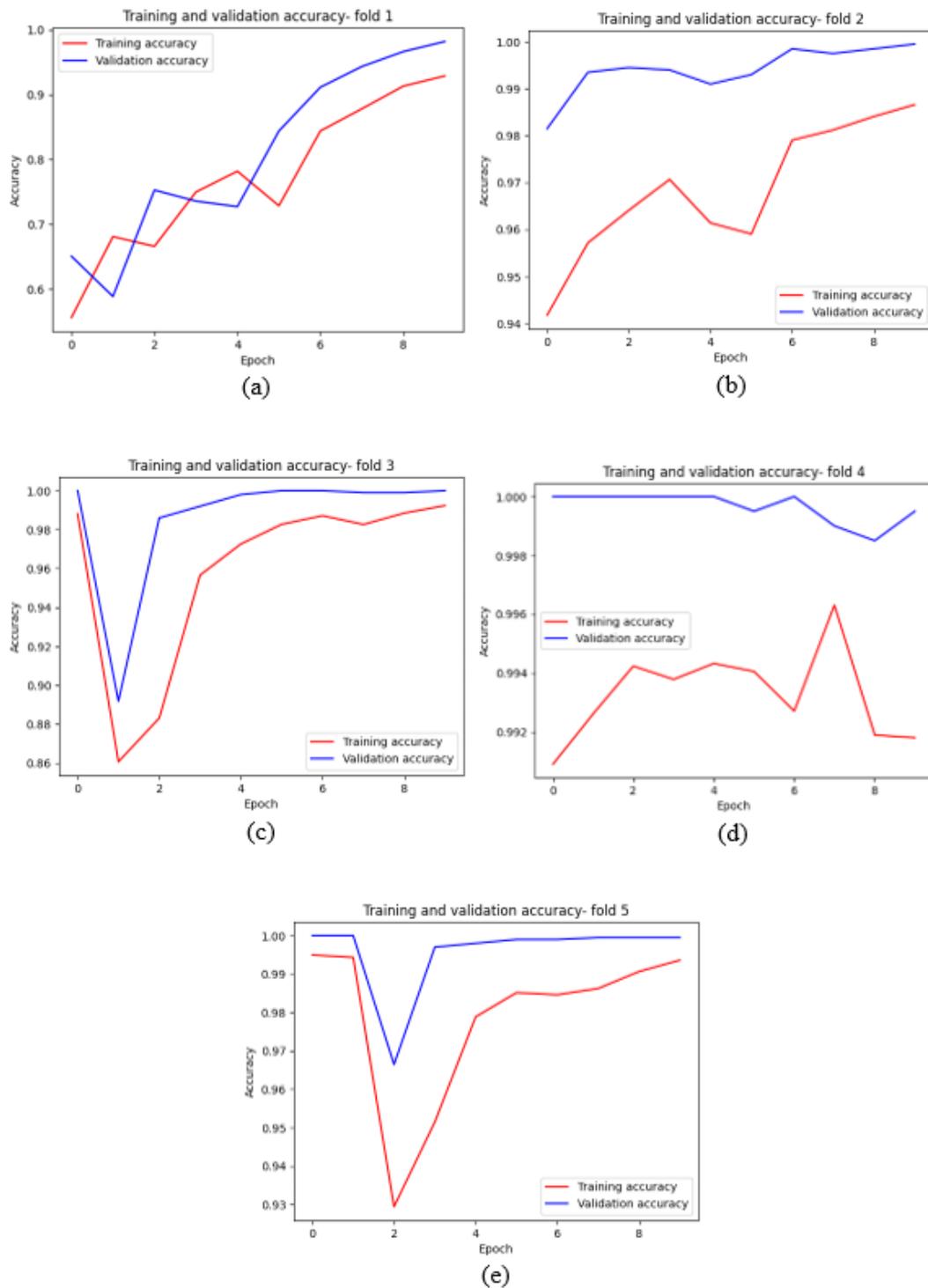


Figure 4-2: Training and Validation Accuracy against Number of Epochs for CNN-AlexNet in (a) First Fold (b) Second Fold (c) Third Fold (d) Fourth Fold (e) Fifth Fold

4.5.1.2 Loss

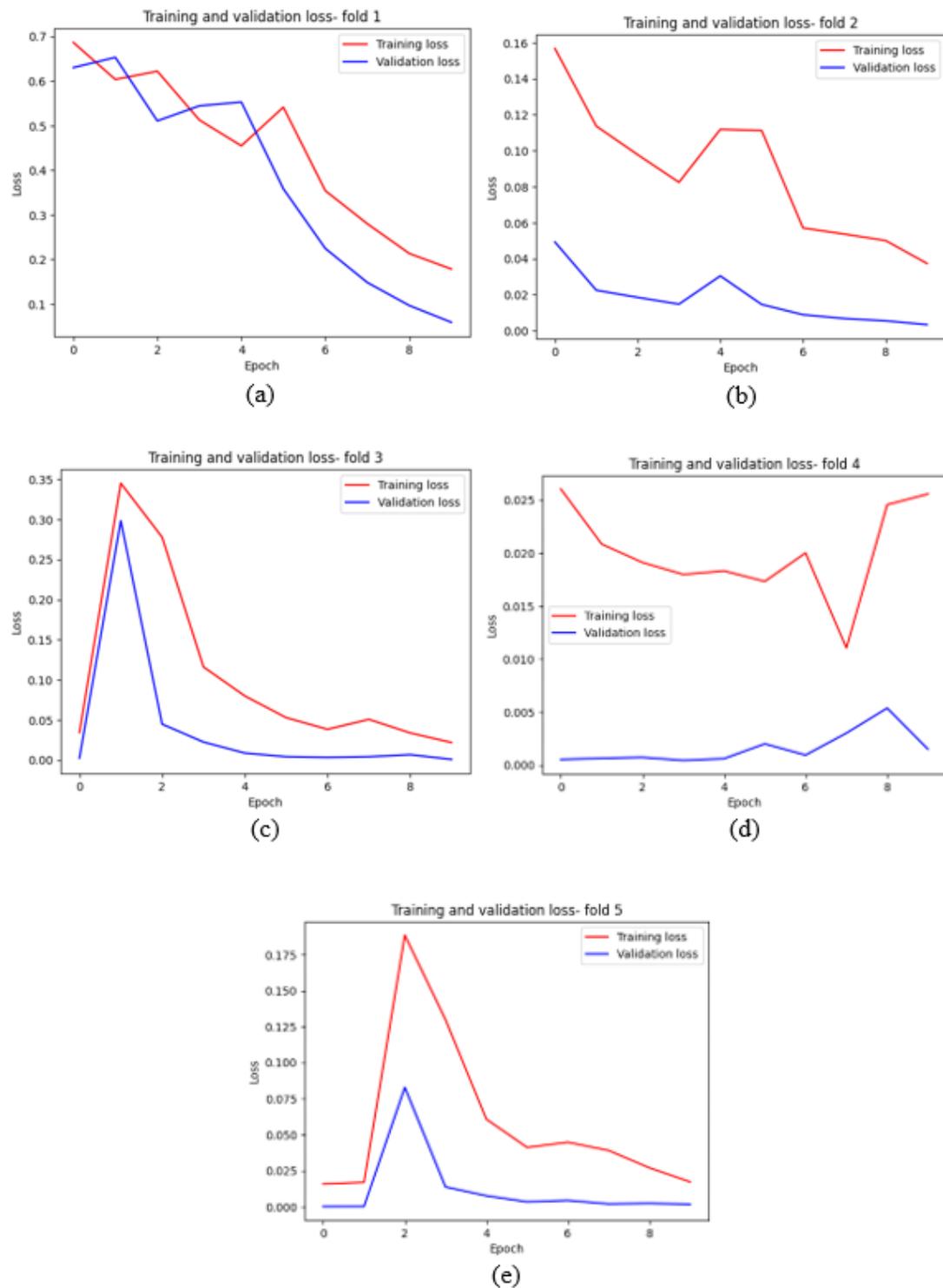


Figure 4-3: Training and Validation Loss against Number of Epochs for CNN-AlexNet in (a) First Fold (b) Second Fold (c) Third Fold (d) Fourth Fold (e) Fifth Fold

4.5.1.3 Confusion Matrix

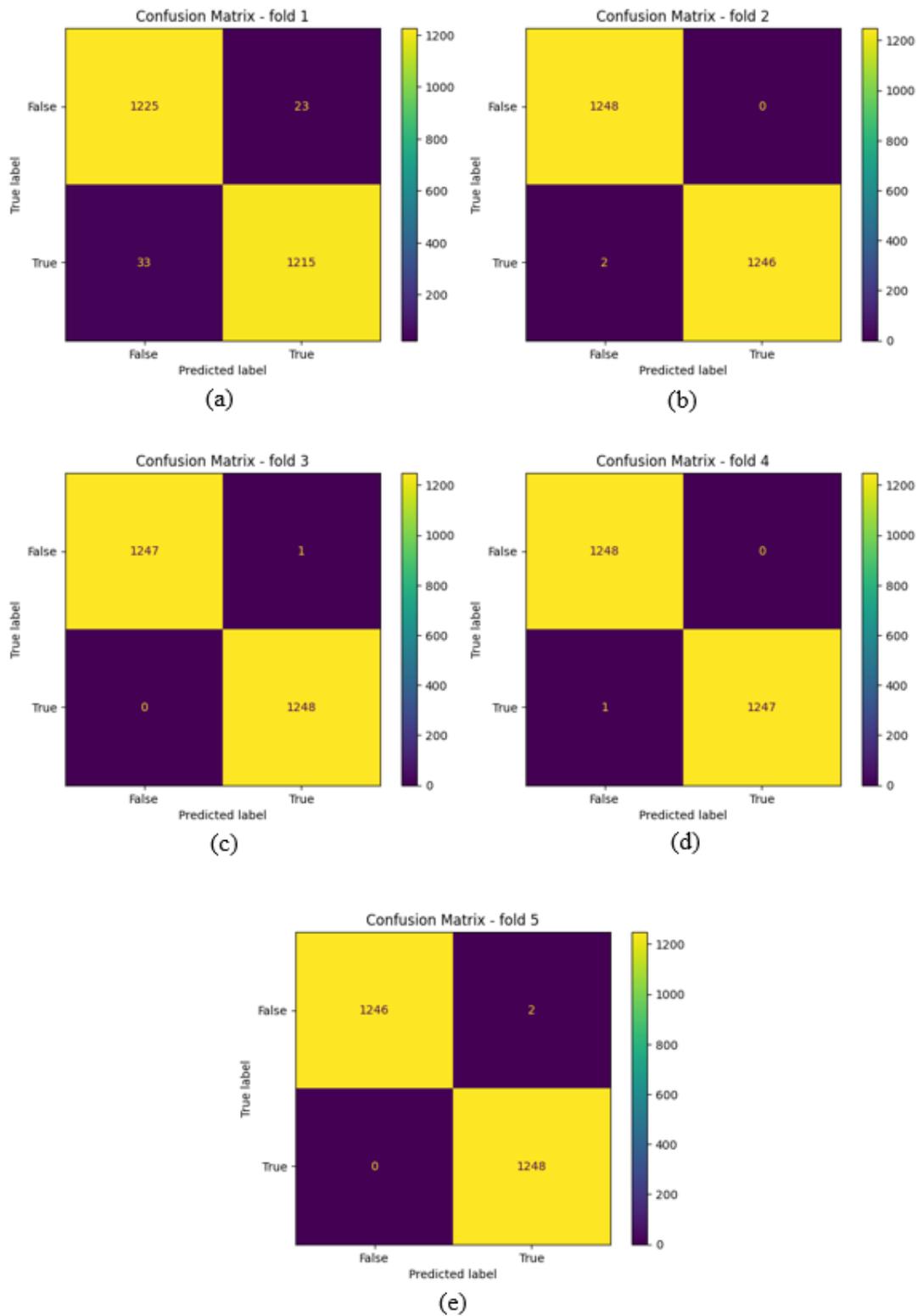


Figure 4-4: Confusion Matrix graph for CNN-AlexNet in (a) First Fold (b) Second Fold (c) Third Fold (d) Fourth Fold (e) Fifth Fold

4.5.2 TL-Inception-V3 with 3 extra hidden layers + dropout

The training process took 10 epochs and 5-fold cross validation. The average training duration of one epoch is approximately 2 minutes and the total training time for 10 epochs and 5-fold cross validation process is approximately 1.7 hours. The TL-Inception-V3 with 3 extra hidden layers + dropout architecture achieved an average validation accuracy of 93.46%; average validation loss of 0.1833; average validation precision of 0.9238; average validation recalls of 0.9605; average validation F1-Score of 0.9413.

The complete 5-fold training results of TL-Inception-V3 with 3 extra hidden layers + dropout model is attached in the Appendix A section below.

Table 21: Validation Evaluation Metrics for TL-Inception-V3 with 3 extra hidden layers + dropout in Each Fold

No of Folds	Accuracy	Loss	Precision	Recall	F1-Score
1	85.76%	0.3500	0.8418	0.9679	0.9005
2	93.46%	0.2033	0.9266	0.9503	0.9383
3	94.88%	0.1492	0.9375	0.9623	0.9497
4	96.13%	0.1226	0.9536	0.9543	0.9539
5	97.05%	0.0912	0.9595	0.9679	0.9639
Average	93.46%	0.1833	0.9238	0.9605	0.9413

4.5.2.1 Accuracy

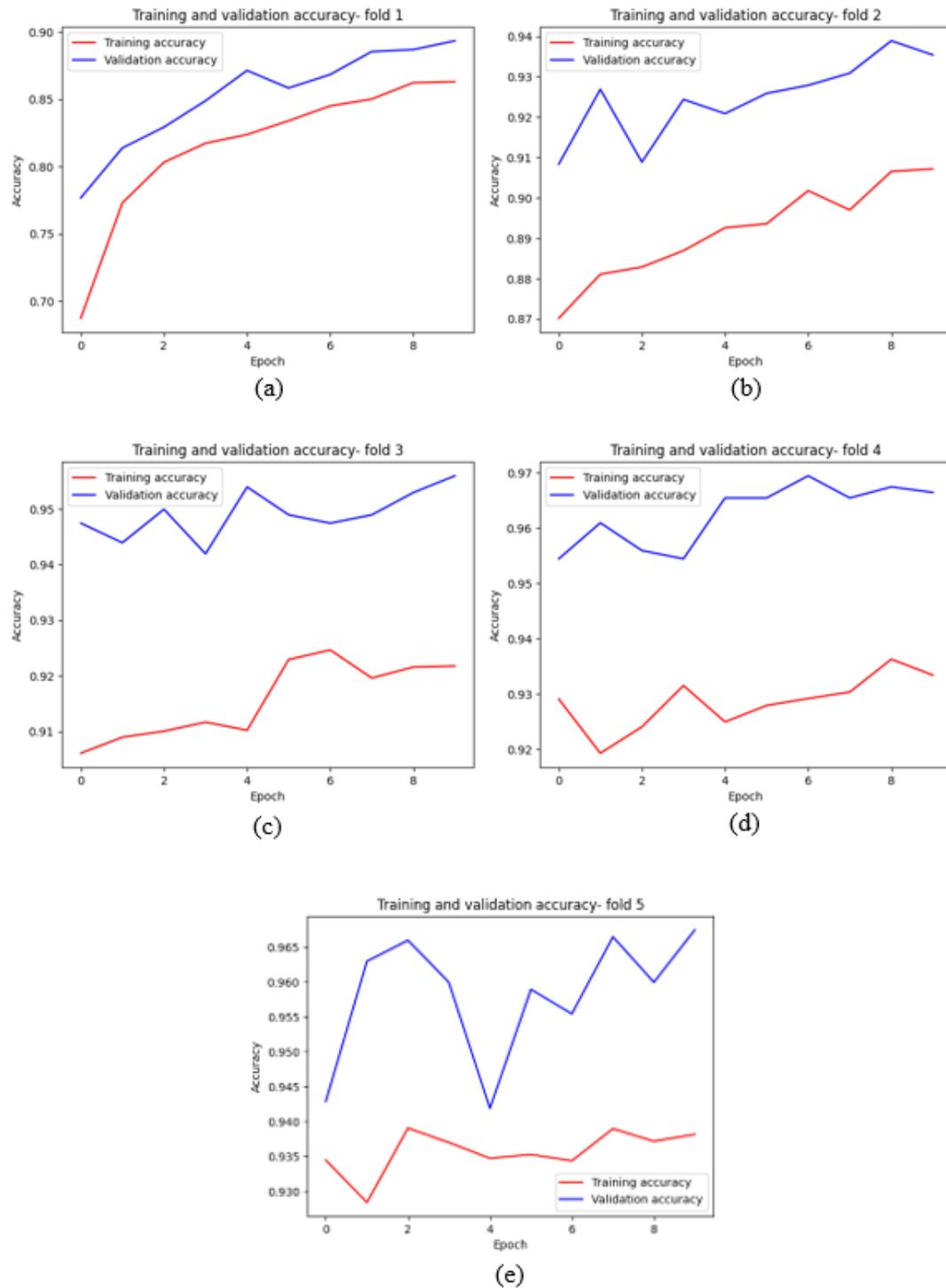


Figure 4-5: Training and Validation Accuracy against Number of Epochs for TL-Inception-V3 with 3 extra hidden layers + dropout in (a) First Fold (b) Second Fold (c) Third Fold (d) Fourth Fold (e) Fifth Fold

4.5.2.2 Loss

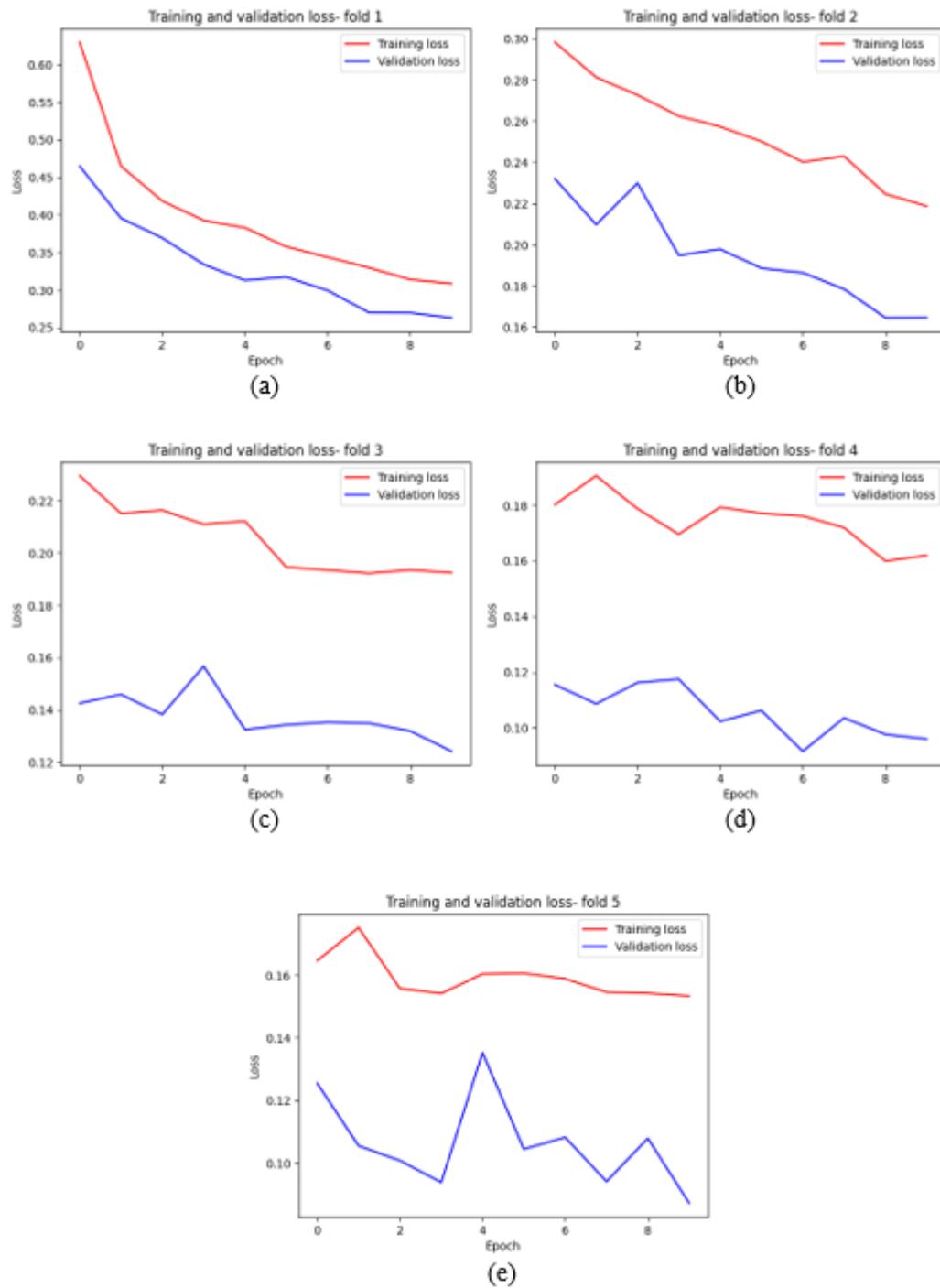


Figure 4-6: Training and Validation Loss against Number of Epochs for TL-Inception-V3 with 3 extra hidden layers + dropout in (a) First Fold (b) Second Fold (c) Third Fold (d) Fourth Fold (e) Fifth Fold

4.5.2.3 Confusion Matrix

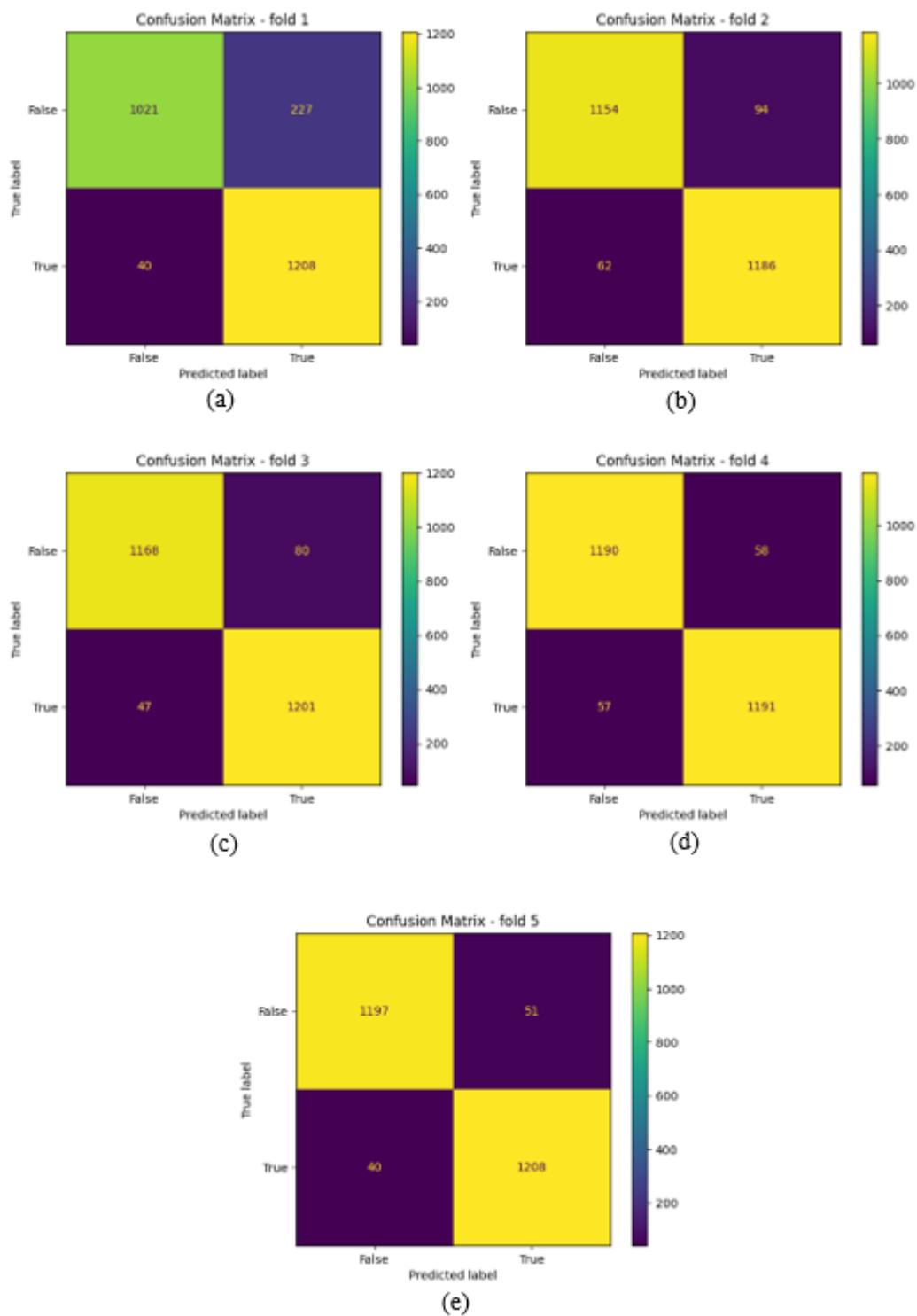


Figure 4-7: Confusion Matrix graph for TL-Inception-V3 with 3 extra hidden layers + dropout in (a) First Fold (b) Second Fold (c) Third Fold (d) Fourth Fold (e) Fifth Fold

4.5.3 TL-DenseNet with 6 extra hidden layers + dropout

The training process took 10 epochs and 5-fold cross validation. The average training duration of one epoch is approximately 2 minutes and the total training time for 10 epochs and 5-fold cross validation process is approximately 1.7 hours. The TL-DenseNet with 6 extra hidden layers + dropout architecture achieved an average validation accuracy of 97.61%; average validation loss of 0.0625; average validation precision of 0.9896; average validation recalls of 0.9883; average validation F1-Score of 0.9889.

The complete 5-fold training results of TL- DenseNet with 6 extra hidden layers + dropout model is attached in the Appendix A section below.

Table 22: Validation Evaluation Metrics for TL-DenseNet with 6 extra hidden layers + dropout in Each Fold

No of Folds	Accuracy	Loss	Precision	Recall	F1-Score
1	91.81%	0.2035	0.9759	0.9720	0.9739
2	97.77%	0.0593	0.9857	0.9920	0.9888
3	99.14%	0.0254	0.9943	0.9848	0.9895
4	99.47%	0.0168	0.9952	0.9992	0.9972
5	99.87%	0.0076	0.9968	0.9936	0.9952
Average	97.61%	0.0625	0.9896	0.9883	0.9889

4.5.3.1 Accuracy

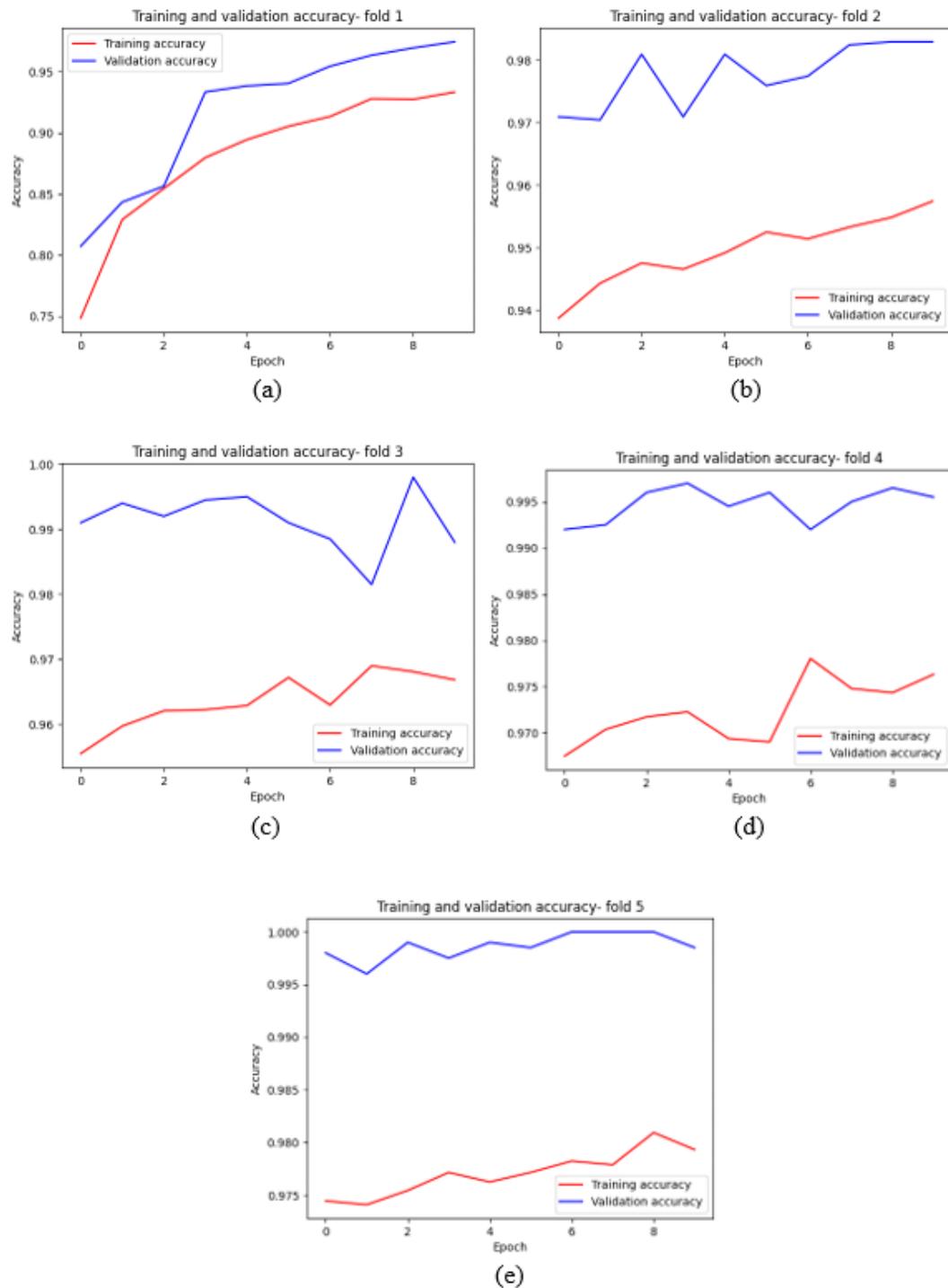


Figure 4-8: Training and Validation Accuracy against Number of Epochs for TL-DenseNet with 6 extra hidden layers + dropout in (a) First Fold (b) Second Fold (c) Third Fold (d) Fourth Fold (e) Fifth Fold

4.5.3.2 Loss

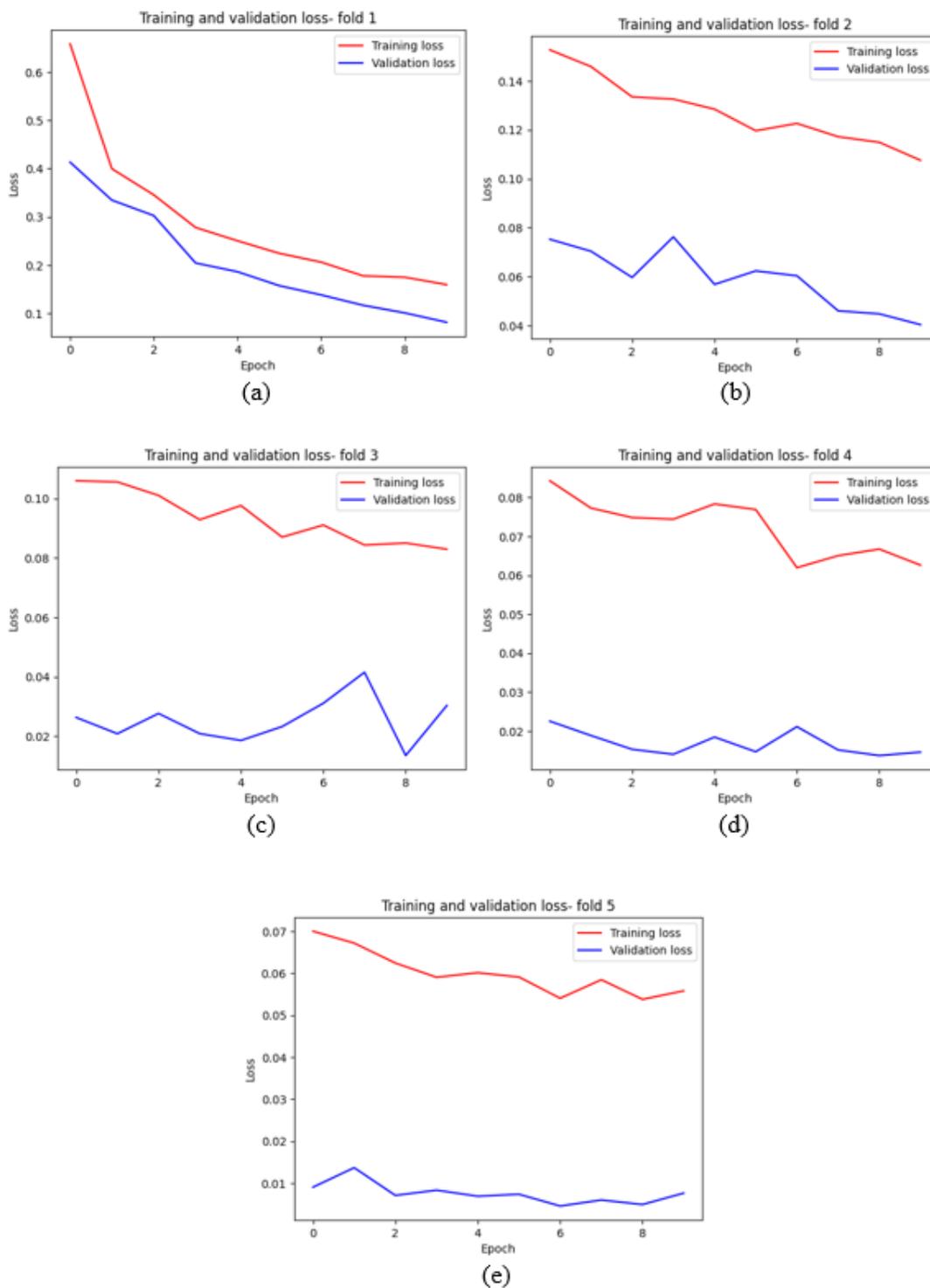


Figure 4-9: Training and Validation Loss against Number of Epochs for TL-DenseNet with 6 extra hidden layers + dropout in (a) First Fold (b) Second Fold (c) Third Fold (d) Fourth Fold (e) Fifth Fold

4.5.3.3 Confusion Matrix

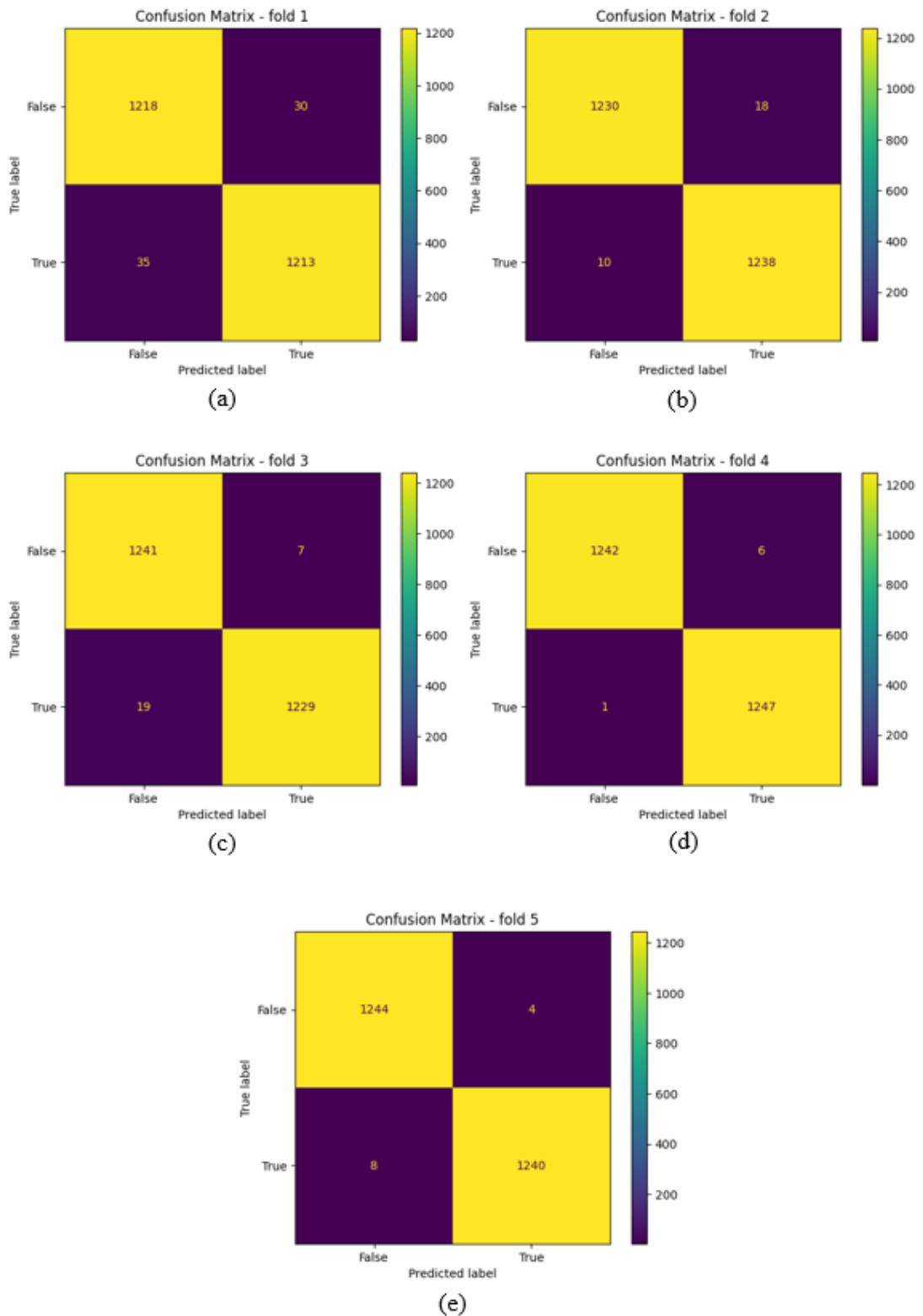


Figure 4-10: Confusion Matrix graph for TL- DenseNet with 6 extra hidden layers + dropout in (a) First Fold (b) Second Fold (c) Third Fold (d) Fourth Fold (e) Fifth Fold

4.6 Testing Results on BUSI dataset

According to *Table 23* below, all three proposed models, CNN-AlexNet model, TL-Inception-V3 with 3 extra hidden layers + dropout model and TL-DenseNet with 6 extra hidden layers + dropout model successfully produced all evaluation metrics, accuracy, precision, recall and F1-score rates above 90%. As illustrated in *Figure 4-12*, the bar chart illustrates the comparison of the proposed models on the testing dataset (BUSI), the TL-DenseNet model performs best in terms of accuracy and F1-score at 91.46% and 0.9144 respectively.

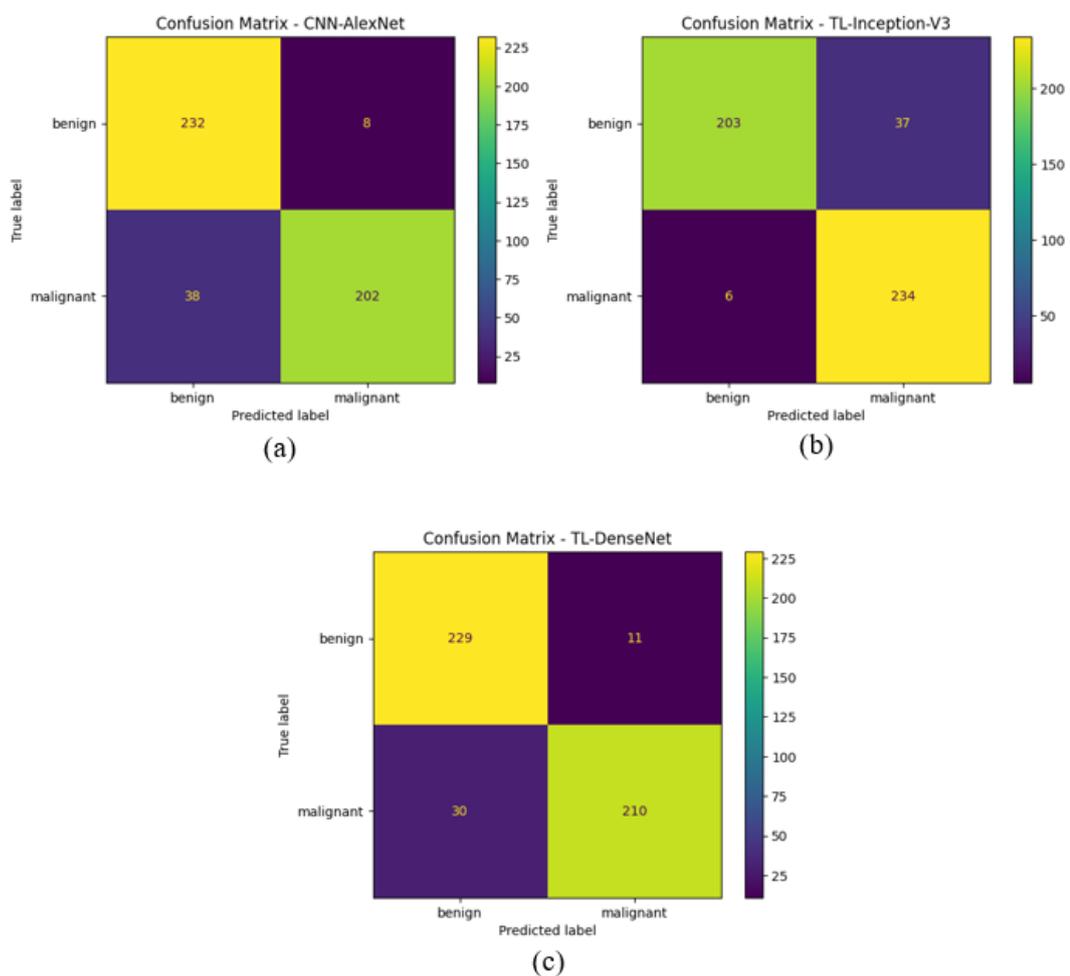


Figure 4-11: Confusion Matrix Graph for (a) CNN-AlexNet (b) TL-Inception-V3 (c) TL-DenseNet on the BUSI dataset

Table 23: Comparison between proposed model in terms of accuracy, precision, recall and F1-score on the BUSI testing dataset

Model	Accuracy	Precision	Recall	F1-Score
CNN-AlexNet	90.42%	0.9106	0.8971	0.9038
TL-Inception-V3	91.04%	0.9174	0.9027	0.9100
TL-DenseNet	91.46%	0.9172	0.9116	0.9144

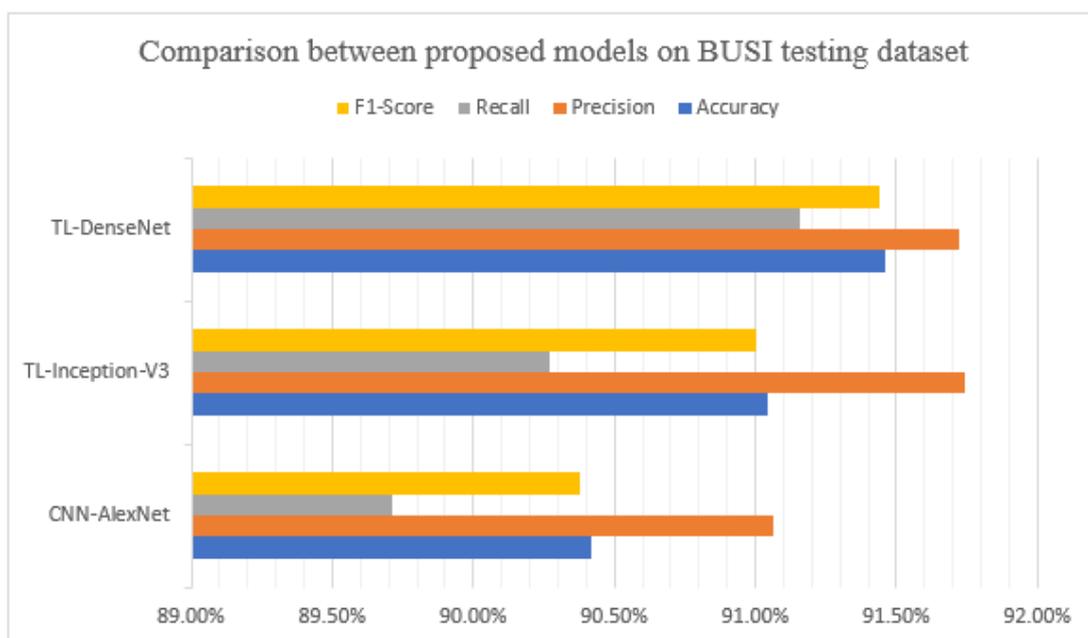


Figure 4-12: Bar chart of comparison between proposed model on BUSI testing set

4.7 Comparison between Existing Techniques

As tabulated in *Table 24* below, the proposed models implemented in this project are compared to existing techniques applied on related works based on the literature review. As observed, all proposed methods, CNN-AlexNet model, TL-Inception-V3 and TL-DenseNet were capable to obtain a high testing accuracy at 90.42%, 91.04% and 91.46% respectively. Among the existing techniques, DCGAN + TL-NASNet model proposed by Al-Dhabyani and TL-Inception-V3 + NN proposed by Gupta achieve the highest and second highest average testing accuracy at 92.6% and 92% respectively. However, this is not an accurate comparison, since these literatures did

not provide the evaluation metrics for precision, recall and F1-score rates. On the other hand, it is worth mentioned that the proposed models in this work uses the MBU dataset + DCGAN with augmentation as training dataset, while the testing results are evaluated on BUSI dataset which is from a completely different source, hence the results are very difficult to achieve higher accuracy due to the distinct configuration such as lighting, ultrasound device, operators and etc. of the different datasets. Therefore, the proposed models are capable to classify the benign and malignant tumours accurately even on unseen datasets.

Table 24: Comparison between proposed models and existing techniques

Proposed Model	Accuracy	Precision	Recall	F1-Score	Train Dataset	Test Dataset
DCGAN +CNN-AlexNet	90.42%	0.9106	0.8971	0.9038	MBU	BUSI
DCGAN + TL-Inception-V3	91.04%	0.9174	0.9027	0.9100		
DCGAN + TL-DenseNet	91.46%	0.9172	0.9116	0.9144		
Existing Techniques						
DCGAN + CNN (Desai et al., 2020)	87%	-	-	-	DDSM	
DCGAN + TL-NASNet (Al-Dhabyani et al., 2019)	92%	-	-	-	BUSI	
CNN (Latif et al., 2019)	88%	-	-	-	MBU	
Quadratic SVM w/ BGWO (Khanna et al., 2021)	84.9%	-	-	-	BUSI	
TL-Inception-V3 + NN (Gupta et al., 2022)	92.6%	-	-	-	BUSI	

Figure 4-13 below illustrates the bar chart of comparison between proposed models and existing techniques based on literature review.

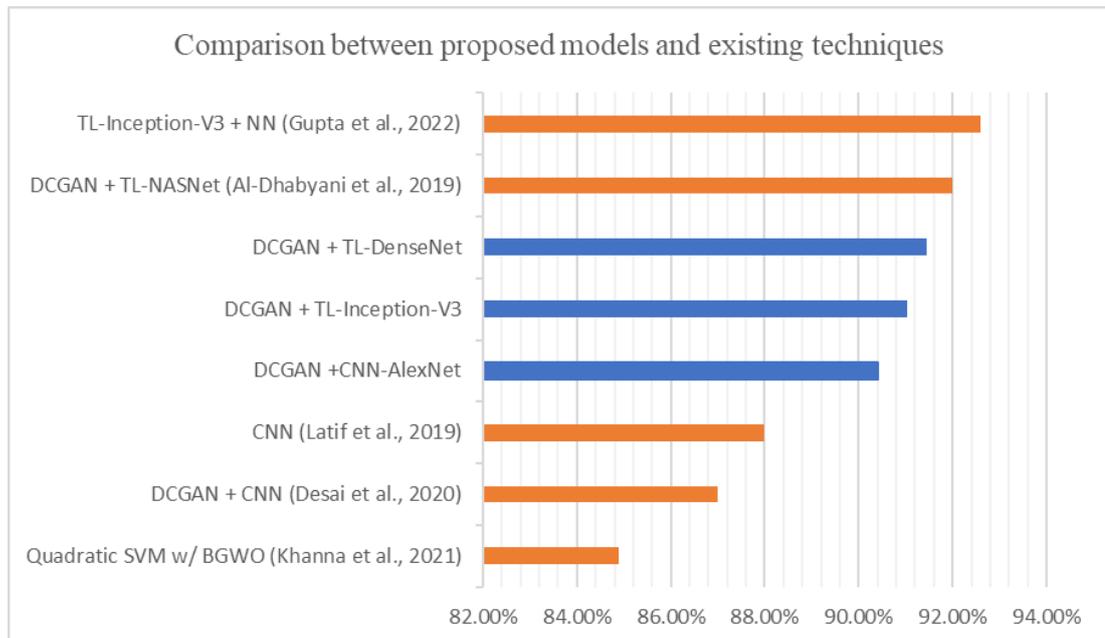


Figure 4-13: Bar chart of comparison between proposed model and existing techniques in terms of accuracy

4.8 Discussion

In recent years, one of the most frequently used imaging technologies in clinical practise is the ultrasound imaging. The ultrasound imaging is considered as a dynamically developing technology with numerous advantages and it has been acknowledged as a potent and commonplace screening and diagnostic tool for clinical research practise. Especially, due to its overall reasonable cost, operator expertise and its relative lower impact to human health, therefore in certain circumstances the ultrasound imaging technology is being favoured compared to CT, MRI and X-Ray. Besides, the ultrasound imaging technology has been widely implemented in the fields of breast diagnostics. Nevertheless, the ultrasound imaging technology also comes with several major drawbacks, for instance image's noises generated by the ultrasound imaging method could significantly affects the image

quality, hence the extensive experience of the diagnostician is heavily relied in order to diagnosis the image accurately (Liu et al., 2019).

On the other hand, the machine learning, image processing techniques and machine vision as lately emerged as the most effective machine learning technology. It has been demonstrated that these strategies can overcome the obstacles of the conventional techniques employed in current industrial imaging technologies. Furthermore, the image processing methods and deep learning algorithms have a strong potential to integrate with the ultrasound imaging technologies in order to contribute in present medical images diagnosis by performing various automated tasks.

In this project, several image processing techniques are implemented on both training and testing ultrasound breast tumours images datasets in order to improve the training process efficiency of the DCGANs and CNNs model. A denoise smoothing filter is chosen and applied to remove the ultrasound image noises while maintaining the tumour edges. Apart from improving the deep learning model efficiency, the pre-processed image could simplify the diagnostics process for people unfamiliar to medical diagnostics to analyse and identify the tumours images.

Since the deep learning algorithm requires numerous of data to fine-tune the parameters of the algorithm after every iteration. This fine-tuning process requires a huge dataset in order to improve the performance of the neural net. Unfortunately, the downloaded datasets from Mendeley website consists of 100 benign and 150 malignant images, which is a very tiny and unbalanced dataset and it could easily overfitting the training model. Therefore, the image augmentation methods are implemented on the original dataset to increase the dataset quantity. The augmentation methods include flipping, rotating, resizing and cropping; thus, the deep learning classifier could train on different orientation of the tumours images in order to allow the algorithm to classify accurately on different variety or perspectives of tumours dataset.

DCGANs is proposed in this project to synthesized realistic benign and malignant breast tumours images. The DCGANs acts as a potentially useful

technique in order to overcome the issue of limited labelled data for the classifier model to classify the breast tumours. Besides, the DCGANs also used for balancing the distribution of dataset, where the minority data class (benign data) is increased to have the same amount as the majority data class. The balanced dataset could greatly improve the performance of the proposed CNNs classifier algorithm. Figure 4-1 below depicted the dataset distribution before and after applying DCGANs.

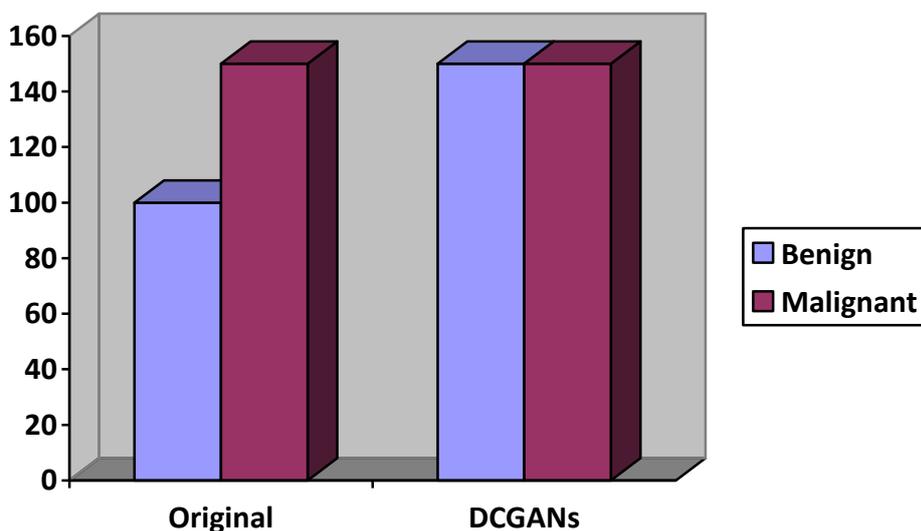


Figure 4-14: Sample distribution of the dataset before and after applying DCGAN

Due to the robust design of the DCGAN, the model could study and train higher hierarchical features and extract useful information from the data rapidly and efficiently. Apart from that, the structure of a normal GANs model consists of fully connected neurons, thus the generated synthesized images are often poor resolution and consists a great ratio of image's noise. Nonetheless due to the stable architecture of DCGANs, it could generate higher quality synthesized images in a shorter duration compared to the basic GANs model. The DCGANs network has been considerably aided in its training by the addition of a batch normalisation layer, which normalises the intermediate input values and speed up the training process.

Table 25: Samples of DCGANs synthesized image with and without batch normalisation layer at 700 training epochs

	Condition	Synthesized image
1	With batch normalisation layer at 700 epochs	
2	Without batch normalisation layer at 700 epochs	

The MBU dataset with implementation of DCGANs was further split randomly into training, validation and testing set. A cross-validation method known as stratified k-fold cross validation ($k=5$) is implemented to the model in order to allow the model to train and validate on 5 different set of training and validation, and therefore enhanced the algorithm. The testing set of MBU dataset is used to evaluate the proposed models on the unseen training set in the first evaluating stage of the project. After the proposed models are capable to perform on the MBU dataset, the classifier is then evaluated on another unseen dataset, BUSI dataset which is obtained from another source.

Based on results in terms of accuracy and F1-score of the proposed CNN-AlexNet model, TL-Inception-V3 with 3 extra hidden layers + dropout model and TL-DenseNet with 6 extra hidden layers + dropout model has achieved 90% and above on training and validation sets. Moreover, all proposed models have performed remarkably and successfully attain above 90% in terms of accuracy and F1-Score on the MBU and BUSI testing datasets. Among the proposed models, the DCGANs with augmentation + TL-DenseNet with 6 extra hidden layers + dropout classifier accomplishes the best performance, accuracy at 91.46% and F1-Score at 0.9144 on the BUSI dataset. Followed by TL-Inception-V3 with 3 extra hidden layers + dropout classifier, which accuracy at 91.04% and F1-Score at 0.91 and CNN-AlexNet model, which accuracy at 90.42% and F1-Score at 0.9038.

In the medical sectors, the deep learning classifier evaluation metrics of precision, recall and F1-score are relatively important compared to the accuracy rates. Undoubtedly, the performance of a medical deep learning classifier is finer if the F1-score rate is higher. Besides, the recall rate is considered as the paramount metrics when classifying cancer, due to its characteristics that quantifies the true positive predictions out of total positive predictions. This is because a cancer classifier with low recall rates may misdiagnosed a cancer positive patient as negative, thus the patient might miss out the ideal opportunity for treatment. In a nutshell, the F1-score is the best evaluation metrics to be considered in medical classification model, due to its mathematical equations that integrated the precision and recall rates. Therefore, without a doubt that the model performs better if the F1-score is approaching 1.00.

After evaluating the testing results, the TL-DenseNet with 6 extra hidden layers + dropout algorithm is the best performed classifier among all proposed models. However, the TL-DenseNet before fine-tuning achieve a testing accuracy at 62.5% and F1-score at 0.6242. *Figure 4-15(a)* illustrates the output statement of the TL-DenseNet before fine-tuning and still utilize the ‘sigmoid’ as output activation function. Therefore, the architecture fine-tuned process is considered as a success endeavour, since a significant improvement is accomplished in the testing results. Similar attempt is applied on the TL-Inception-V3 model, *Figure 4-15(b)* shows the output statement of TL-Inception-V3 model before fine-tuned. The testing accuracy TL-Inception-V3 model before fine-tuned is 48.13% and F1-score at 0.4191, which the performance is poor and unacceptable. Nevertheless, the performance of the model after fine-tuned had improved in a significant way.

<pre> =====TEST RESULTS===== Found 480 images belonging to 2 classes. 15/15 [=====] - 13s 658ms/step Accuracy : 0.625 Precision : 0.6260592478464878 f1Score : 0.624210581255763 [[139 101] [79 161]] </pre>	<pre> =====TEST RESULTS===== Found 480 images belonging to 2 classes. 15/15 [=====] - 6s 249ms/step Accuracy : 0.48125 Precision : 0.46722405996783095 f1Score : 0.41910367385820724 [[194 46] [203 37]] </pre>
(a)	(b)

Figure 4-15: Output statement of (a) TL-DenseNet and (b) TL-Inception-V3 before fine-tuned

CHAPTER 5

CONCLUSION AND RECOMMENDATIONS

5.1 Project Review

The main purpose of this project is to develop an image processing with deep learning model to classify and detect breast tumours ultrasound images. The first objective of the project is to design an image processing method to reduce the image noise of the ultrasound images of benign and malignant tumours. Therefore, the denoise smoothing filter is applied to the ultrasound images using OpenCV. The noises in the ultrasound images are reduced while maintaining the edges of the tumours, thus the objective is achieved.

The second objective is to apply data augmentation methods, such as GANs to increase the dataset quantity. Thus, a DCGANs model is developed accordingly to the research paper by Radford (2016) to generate synthesized realistic breast tumours images. Moreover, several data augmentation techniques, for instance flipping, rotating, resizing and cropping images have been applied to the dataset to enhance the data quality and increase the data quantity, therefore the objective is attained.

The third and fourth objective is to design a CNN-based classifier to classify benign and malignant tumors and apply a suitable transfer learning model and tune the model's parameters in order to improve the accuracy of the classifier. Hence, three deep learning models which includes the CNN-AlexNet, TL-Inception-V3 and TL-DenseNet were proposed in this project. Furthermore, the TL-Inception-V3 and TL-DenseNet models were fine-tuned by adding several hidden layers such as fully

connected layer, batch normalization layer with dropout and replace the sigmoid output activation function in order to improve the algorithm performance. In a nutshell, all objectives in this project have been accomplished.

5.2 Project Findings

The transfer learning method in deep learning is defined as the approach of utilizing a pre-trained model and the architecture is designed by the deep neural networks related company, therefore the architecture of the transfer learning models is well refined by deep learning scientist with proven outstanding results and strategies. Besides, the parameters and weights of the transfer learning models are trained on a gigantic number of datasets consisting of various features and the training process often requires several high-powered GPUs for a long period of time since training on large datasets are time consuming. On the other hand, in this project, among all proposed models, before the fine-tuning technique is implemented, the CNN-AlexNet performs best in terms of accuracy and F1-score. While the transfer learning approach were underperformance where the TL-DenseNet achieved accuracy around 62% and TL-Inception-V3 obtained accuracy of approximately 48%. The presumption is the transfer learning models has not trained on the tumours-related data. Additionally, the last layer of transfer learning models is the classification layer that often used to predict on the pre-trained dataset related image. Therefore, if the last few layers of the transfer learning models are not freeze, the deep learning classifier is predicting on data which has not trained previously. Furthermore, the transfer learning models generally deep and consist of numerous layers, thus if the earlier layers did not set to non-trainable, the tumours data will be faded out thorough the numerous layers before arriving the final classifier layer and the training process will be more time consuming. According to Vinithavn (2021), the earlier layer tends to capture more generic features, while the later layers are more dataset specific. Therefore, the approach in this project is to set the transfer learning model to non-trainable and attach several layers for training purpose such as fully connected layers and batch normalization layers with dropout function. This fine-tuning technique could freeze the transfer learning model and trained on the designed hidden layers;

therefore, the model could utilize the generic features of the transfer learning model and able to train on the prepared dataset simultaneously. Moreover, the last classification layer of the transfer learning model is removed and replaced by the proposed classification layer.

5.3 Recommendations for Future Improvement

First and foremost, this project has successfully developed a fine-tuned TL-Inception-V3 and TL-DenseNet model with the implementation of DCGANs with image augmentation and processing methods for classifying benign and malignant breast tumors ultrasound images and capable to accomplish an outstanding performance in terms of accuracy and F1-score. However, due to the project period, I was unable to test the CNN algorithm with different layers and activation function. Therefore, the suggestion is to refine the CNN algorithm by adding or replacing suitable layers with various dropout value and test with another suitable activation function experimentally. In addition, different transfer learning models are also suggested to test on these datasets experimentally.

Apart from that, the data augmentation part of the project with the implementation of DCGANs is the most time-consuming part of the project, almost 70% of the overall project duration has been used on training and generating synthesized images using the DCGANs model. Therefore, the recommendation is to study on various GANs model in order to fine-tune the DCGANs architecture for shortening the training duration and improve the performance of DCGANs in terms of generating synthesized tumors images. Moreover, the Pix2Pix technique is recommended for data augmentation by generating synthesized image due to its characteristics of image-to-image translation, thus the fake tumor images can be synthesized from free-form sketch. Figure 5-1 illustrates the outline of the implementation of Pix2Pix technique to generate lung cancer CT image.

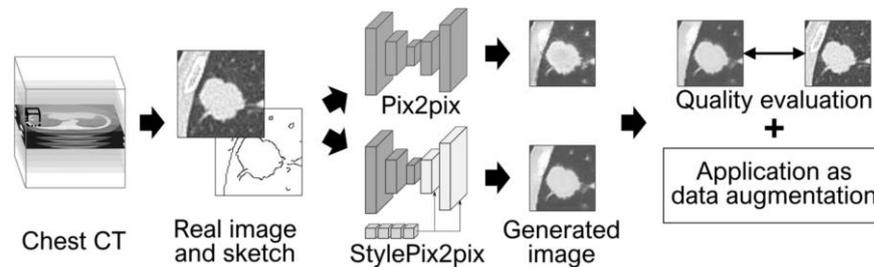


Figure 5-1: Outline of the implementation of Pix2Pix technique to generate lung cancer CT image (Toda et al., 2022)

5.4 Conclusion

The breast cancer is life-threatening, and it is also one of the leading causes of death, therefore early diagnosis of breast cancer acts as an important role in order to prevent the cancer by progressing rapidly and starting to affect human's health condition or even worse, approaching death. The early diagnosis process could allow doctors to provide treatments and operations that could end up saving the patients' lives. This paper proposed three CNN models including transfer learning with integration of DCGANs for data augmentation and image processing methods to classify the breast tumors as benign and malignant types. The Mendeley Breast Ultrasound dataset was used to train and validate the proposed deep learning classifier model and the Breast Ultrasound Image dataset was used to test the accuracy of the classifier in classifying benign and malignant tumors. Furthermore, the image processing methods have been implemented on the datasets to remove the ultrasound noises and thus enhance the image quality. Moreover, DCGANs model has successfully generate synthesized both benign and malignant breast tumors ultrasound images and image augmentation techniques such as flipping and rotating images have successfully increase the dataset quantity. Apart from that, the proposed models, CNN-AlexNet, TL-Inception-V3 and TL-DenseNet have successfully developed and able to classify the tumors images accurately with a testing accuracy at 90.42%, 91.04% and 91.46% and F1-score at 0.9038, 0.9100 and 0.9144 respectively. Without a doubt, among the three proposed models, the fine-tuned TL-DenseNet exhibited the finest performance, followed by the fine-tuned TL-Inception-V3. In a nutshell, all objectives of this project are accomplished.

REFERENCES

- Abdullah, N.A., Mahiyuddin, W.R.W., Muhammad, N.A., Ali, Z.M., Ibrahim, L., Tamim, N.S.I., Mustafa, A.N. and Kamaluddin, M.A. (2013). Survival Rate of Breast Cancer Patients In Malaysia: A Population-based Study. *Asian Pacific Journal of Cancer Prevention*, 14(8), pp.4591–4594. doi:10.7314/apjcp.2013.14.8.4591.
- Al-Dhabyani, W., Gomaa, M., Khaled, H. and Fahmy, A. (2019). Deep Learning Approaches for Data Augmentation and Classification of Breast Masses using Ultrasound Images. *International Journal of Advanced Computer Science and Applications (IJACSA)*, [online] 10(5). doi:10.14569/IJACSA.2019.0100579.
- Al-Dhabyani, W., Gomaa, M., Khaled, H. and Fahmy, A. (2020). Dataset of breast ultrasound images. *Data in Brief*, 28, p.104863. doi:10.1016/j.dib.2019.104863.
- Alyafi, B., Diaz, O. and Martí, R. (2020). DCGANs for realistic breast mass augmentation in x-ray mammography. *Medical Imaging 2020: Computer-Aided Diagnosis*. doi:10.1117/12.2543506.
- Basavarajaiah, M. (2022). *6 basic things to know about Convolution*. [online] Medium. Available at: <https://medium.com/@bdhuma/6-basic-things-to-know-about-convolution-daef5e1bc411#:~:text=In%20image%20processing%2C%20convolution%20is>.
- Brazier, Y. (2019). *Tumors: Benign, premalignant, and malignant*. [online] www.medicalnewstoday.com. Available at: <https://www.medicalnewstoday.com/articles/249141>.
- Breast Cancer Foundation. (n.d.). *About Breast Cancer*. [online] Available at: <https://www.breastcancerfoundation.org.my/about-breast-cancer>.
- Cleveland Clinic. (2021). *Tumor: What Is It, Types, Symptoms, Treatment & Prevention*. [online] Available at: <https://my.clevelandclinic.org/health/diseases/21881-tumor#:~:text=A%20tumor%20is%20a%20solid>.
- Desai, S.D., Giraddi, S., Verma, N., Gupta, P. and Ramya, S. (2020). Breast Cancer Detection Using GAN for Limited Labeled Dataset. *2020 12th International Conference on Computational Intelligence and Communication Networks*

(CICN). doi:10.1109/cicn49253.2020.9242551.

- Dinakaran, R.K., Easom, P., Bouridane, A., Zhang, L., Jiang, R., Mehboob, F. and Rauf, A. (2019). Deep Learning Based Pedestrian Detection at Distance in Smart Cities. *Advances in Intelligent Systems and Computing*, pp.588–593. doi:10.1007/978-3-030-29513-4_43.
- Fang, W., Zhang, F., S. Sheng, V. and Ding, Y. (2018). A Method for Improving CNN-Based Image Recognition Using DCGAN. *Computers, Materials & Continua*, 57(1), pp.167–178. doi:10.32604/cmc.2018.02356.
- Frid-Adar, M., Diamant, I., Klang, E., Amitai, M., Goldberger, J. and Greenspan, H. (2018). GAN-based synthetic medical image augmentation for increased CNN performance in liver lesion classification. *Neurocomputing*, 321, pp.321–331. doi:10.1016/j.neucom.2018.09.013.
- Fung, V. (2017). *An Overview of ResNet and its Variants*. [online] Towards Data Science. Available at: <https://towardsdatascience.com/an-overview-of-resnet-and-its-variants-5281e2f56035>.
- GeeksforGeeks. (2021). *Python OpenCV - Smoothing and Blurring*. [online] Available at: <https://www.geeksforgeeks.org/python-opencv-smoothing-and-blurring/> [Accessed 15 Sep. 2022].
- GeeksforGeeks. (2022). *What is so special about Generative Adversarial Network (GAN)*. [online] Available at: <https://www.geeksforgeeks.org/what-is-so-special-about-generative-adversarial-network-gan/> [Accessed 15 Sep. 2022].
- Global Cancer Observatory (2020). *Global Cancer Observatory*. [online] Iarc.fr. Available at: <https://gco.iarc.fr/>.
- Goodfellow, I., Pouget-Abadie, J., Mirza, M., Xu, B., Warde-Farley, D., Ozair, S., Courville, A. and Bengio, Y. (2020). Generative adversarial networks. *Communications of the ACM*, 63(11), pp.139–144. doi:10.1145/3422622.
- Gupta, S., Panwar, A., Yadav, R., Aeri, M. and Manwal, M. (2022). Employing Deep Learning Feature Extraction Models with Learning Classifiers to Diagnose Breast Cancer in Medical Images. *2022 IEEE Delhi Section Conference (DELCON)*. doi:10.1109/delcon54057.2022.9752856.
- He, K., Zhang, X., Ren, S. and Sun, J. (2016). Deep Residual Learning for Image Recognition. *2016 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pp.770–778. doi:10.1109/cvpr.2016.90.
- Hiremath, P.S., T., P. and Badiger, S. (2013). Speckle Noise Reduction in Medical Ultrasound Images. *Advancements and Breakthroughs in Ultrasound Imaging*. doi:10.5772/56519.
- Hochreiter, S. and Schmidhuber, J. (1997). Long Short-Term Memory. *Neural Computation*, 9(8), pp.1735–1780. doi:10.1162/neco.1997.9.8.1735.
- Iqbal, T. and Ali, H. (2018). Generative Adversarial Network for Medical Images

- (MI-GAN). *Journal of Medical Systems*, 42(11). doi:10.1007/s10916-018-1072-9.
- Irla, T. (2019). *Transfer Learning using Inception-v3 for Image Classification*. [online] Analytics Vidhya. Available at: <https://medium.com/analytics-vidhya/transfer-learning-using-inception-v3-for-image-classification-86700411251b>.
- Jason Brownlee (2019). *A Gentle Introduction to Generative Adversarial Networks (GANs)*. [online] Machine Learning Mastery. Available at: <https://machinelearningmastery.com/what-are-generative-adversarial-networks-gans/>.
- Kazemina, S., Baur, C., Kuijper, A., van Ginneken, B., Navab, N., Albarqouni, S. and Mukhopadhyay, A. (2020). GANs for medical image analysis. *Artificial Intelligence in Medicine*, [online] 109, p.101938. doi:10.1016/j.artmed.2020.101938.
- Khanna, P., Sahu, M. and Kumar Singh, B. (2021). Improving the classification performance of breast ultrasound image using deep learning and optimization algorithm. *2021 IEEE International Conference on Technology, Research, and Innovation for Betterment of Society (TRIBES)*. doi:10.1109/tribes52498.2021.9751677.
- Krizhevsky, A., Sutskever, I. and Hinton, G.E. (2012). ImageNet classification with deep convolutional neural networks. *Communications of the ACM*, 60(6), pp.84–90. doi:10.1145/3065386.
- Kuhl, C.K., Schrading, S., Leutner, C.C., Morakkabati-Spitz, N., Wardelmann, E., Fimmers, R., Kuhn, W. and Schild, H.H. (2005). Mammography, Breast Ultrasound, and Magnetic Resonance Imaging for Surveillance of Women at High Familial Risk for Breast Cancer. *Journal of Clinical Oncology*, 23(33), pp.8469–8476. doi:10.1200/jco.2004.00.4960.
- Kumar, A. (2022). *K-Fold Cross Validation - Python Example*. [online] Data Analytics. Available at: <https://vitalflux.com/k-fold-cross-validation-python-example/> [Accessed 15 Sep. 2022].
- Latif, G., Butt, M.O., Yousif Al Anezi, F. and Alghazo, J. (2020). Ultrasound Image Despeckling and detection of Breast Cancer using Deep CNN. *2020 RIVF International Conference on Computing and Communication Technologies (RIVF)*. doi:10.1109/rivf48685.2020.9140767.
- Liu, S., Wang, Y., Yang, X., Lei, B., Liu, L., Li, S.X., Ni, D. and Wang, T. (2019). Deep Learning in Medical Ultrasound Analysis: A Review. *Engineering*, 5(2), pp.261–275. doi:10.1016/j.eng.2018.11.020.
- National Cancer Institute (2021). *What Is Cancer?* [online] National Cancer Institute. Available at: <https://www.cancer.gov/about-cancer/understanding/what-is-cancer>.
- Our World in Data. (2019). *Number of deaths by cause*. [online] Available at:

<https://ourworldindata.org/grapher/annual-number-of-deaths-by-cause>.

- Radford, A., Metz, L., & Chintala, S. (2016). Unsupervised Representation Learning with Deep Convolutional Generative Adversarial Networks. CoRR, abs/1511.06434.
- Raj, B. (2018). *A Simple Guide to the Versions of the Inception Network*. [online] Towards Data Science. Available at: <https://towardsdatascience.com/a-simple-guide-to-the-versions-of-the-inception-network-7fc52b863202>.
- Rodrigues, Paulo Sergio (2018), “Breast Ultrasound Image”, Mendeley Data, V1, doi: 10.17632/wmy84gzngw.1
- Salimans, T., Goodfellow, I.J., Zaremba, W., Cheung, V., Radford, A., & Chen, X. (2016). Improved Techniques for Training GANs. ArXiv, abs/1606.03498.
- Senaras, C., Niazi, M.K.K., Sahiner, B., Pennell, M.P., Tozbikian, G., Lozanski, G. and Gurcan, M.N. (2018). Optimized generation of high-resolution phantom images using cGAN: Application to quantification of Ki67 breast cancer images. *PLOS ONE*, 13(5), p.e0196846. doi:10.1371/journal.pone.0196846.
- Shin, H.-C., Tenenholtz, N.A., Rogers, J.K., Schwarz, C.G., Senjem, M.L., Gunter, J.L., Andriole, K.P. and Michalski, M. (2018). Medical Image Synthesis for Data Augmentation and Anonymization Using Generative Adversarial Networks. *Simulation and Synthesis in Medical Imaging*, pp.1–11. doi:10.1007/978-3-030-00536-8_1.
- Srivastava, R.K., Greff, K., & Schmidhuber, J. (2015). Highway Networks. ArXiv, abs/1505.00387.
- Szegedy, C., Liu, W., Jia, Y., Sermanet, P., Reed, S., Anguelov, D., Erhan, D. and Vanhoucke, V. (2015). Going deeper with convolutions. *2015 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*. [online] doi:10.1109/cvpr.2015.7298594.
- Szegedy, C., Vanhoucke, V., Ioffe, S., Shlens, J. and Wojna, Z. (2016). Rethinking the Inception Architecture for Computer Vision. *2016 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*. doi:10.1109/cvpr.2016.308.
- Thuy, M.B.H. and Hoang, V.T. (2020). *Fusing of Deep Learning, Transfer Learning and GAN for Breast Cancer Histopathological Image Classification*. [online] Springer Link. doi:10.1007/978-3-030-38364-0_23.
- Toda, R., Teramoto, A., Kondo, M., Imaizumi, K., Saito, K. and Fujita, H. (2022). Lung cancer CT image generation from a free-form sketch using style-based pix2pix for data augmentation. *Scientific Reports*, 12(1). doi:10.1038/s41598-022-16861-5.
- Vinithavn (2021). *The Power Of Transfer Learning in Deep Learning*. [online] Analytics Vidhya. Available at: <https://medium.com/analytics-vidhya/the-power-of-transfer-learning-in-deep-learning-681f86a62f79> [Accessed 15 Sep.

2022].

- Wang, G., Li, W., Ourselin, S. and Vercauteren, T. (2018). Automatic Brain Tumor Segmentation Using Cascaded Anisotropic Convolutional Neural Networks. *Brainlesion: Glioma, Multiple Sclerosis, Stroke and Traumatic Brain Injuries*, pp.178–190. doi:10.1007/978-3-319-75238-9_16.
- Yip, C. H., Taib, N. A., & Mohamed, I. (2006). Epidemiology of breast cancer in Malaysia. *Asian Pacific journal of cancer prevention: APJCP*, 7(3), 369–374.
- Zhi, W., Yueng, H.W.F., Chen, Z., Zandavi, S.M., Lu, Z. and Chung, Y.Y. (2017). Using Transfer Learning with Convolutional Neural Networks to Diagnose Breast Cancer from Histopathological Images. *Neural Information Processing*, pp.669–676. doi:10.1007/978-3-319-70093-9_71.

APPENDICES

APPENDIX A: Training Output Statement of Proposed Models Generated in PyCharm

```

Results for fold 1
Found 11107 images belonging to 2 classes.
Found 1997 images belonging to 2 classes.
Epoch 1/10
44/44 [=====] - 572s 13s/step - loss: 0.6865 - accuracy: 0.5559 - val_loss: 0.6302 - val_accuracy: 0.6505
Epoch 2/10
44/44 [=====] - 547s 12s/step - loss: 0.6036 - accuracy: 0.6808 - val_loss: 0.6531 - val_accuracy: 0.5884
Epoch 3/10
44/44 [=====] - 605s 14s/step - loss: 0.6220 - accuracy: 0.6658 - val_loss: 0.5107 - val_accuracy: 0.7526
Epoch 4/10
44/44 [=====] - 628s 14s/step - loss: 0.5127 - accuracy: 0.7496 - val_loss: 0.5444 - val_accuracy: 0.7356
Epoch 5/10
44/44 [=====] - 615s 14s/step - loss: 0.4547 - accuracy: 0.7818 - val_loss: 0.5529 - val_accuracy: 0.7271
Epoch 6/10
44/44 [=====] - 614s 14s/step - loss: 0.5414 - accuracy: 0.7285 - val_loss: 0.3585 - val_accuracy: 0.8438
Epoch 7/10
44/44 [=====] - 623s 14s/step - loss: 0.3541 - accuracy: 0.8439 - val_loss: 0.2245 - val_accuracy: 0.9114
Epoch 8/10
44/44 [=====] - 624s 14s/step - loss: 0.2800 - accuracy: 0.8781 - val_loss: 0.1482 - val_accuracy: 0.9434
Epoch 9/10
44/44 [=====] - 668s 15s/step - loss: 0.2132 - accuracy: 0.9131 - val_loss: 0.0965 - val_accuracy: 0.9664
Epoch 10/10
44/44 [=====] - 655s 15s/step - loss: 0.1784 - accuracy: 0.9289 - val_loss: 0.0590 - val_accuracy: 0.9820

```

Figure 5-2: Training output statement of CNN-AlexNet in fold 1

```

Results for fold 2
Found 11107 images belonging to 2 classes.
Found 1997 images belonging to 2 classes.
Epoch 1/10
44/44 [=====] - 621s 14s/step - loss: 0.1568 - accuracy: 0.9417 - val_loss: 0.0492 - val_accuracy: 0.9815
Epoch 2/10
44/44 [=====] - 625s 14s/step - loss: 0.1137 - accuracy: 0.9571 - val_loss: 0.0224 - val_accuracy: 0.9935
Epoch 3/10
44/44 [=====] - 630s 14s/step - loss: 0.0978 - accuracy: 0.9641 - val_loss: 0.0185 - val_accuracy: 0.9945
Epoch 4/10
44/44 [=====] - 654s 15s/step - loss: 0.0824 - accuracy: 0.9706 - val_loss: 0.0147 - val_accuracy: 0.9940
Epoch 5/10
44/44 [=====] - 670s 15s/step - loss: 0.1119 - accuracy: 0.9614 - val_loss: 0.0304 - val_accuracy: 0.9910
Epoch 6/10
44/44 [=====] - 679s 15s/step - loss: 0.1112 - accuracy: 0.9590 - val_loss: 0.0145 - val_accuracy: 0.9930
Epoch 7/10
44/44 [=====] - 595s 13s/step - loss: 0.0571 - accuracy: 0.9790 - val_loss: 0.0088 - val_accuracy: 0.9985
Epoch 8/10
44/44 [=====] - 540s 12s/step - loss: 0.0537 - accuracy: 0.9812 - val_loss: 0.0067 - val_accuracy: 0.9975
Epoch 9/10
44/44 [=====] - 590s 13s/step - loss: 0.0500 - accuracy: 0.9841 - val_loss: 0.0054 - val_accuracy: 0.9985
Epoch 10/10
44/44 [=====] - 534s 12s/step - loss: 0.0373 - accuracy: 0.9866 - val_loss: 0.0033 - val_accuracy: 0.9995

```

Figure 5-3: Training output statement of CNN-AlexNet in fold 2

```

Results for fold 3
Found 11107 images belonging to 2 classes.
Found 1997 images belonging to 2 classes.
Epoch 1/10
44/44 [=====] - 523s 12s/step - loss: 0.0341 - accuracy: 0.9880 - val_loss: 0.0022 - val_accuracy: 1.0000
Epoch 2/10
44/44 [=====] - 520s 12s/step - loss: 0.3452 - accuracy: 0.8607 - val_loss: 0.2984 - val_accuracy: 0.8918
Epoch 3/10
44/44 [=====] - 518s 12s/step - loss: 0.2779 - accuracy: 0.8831 - val_loss: 0.0447 - val_accuracy: 0.9860
Epoch 4/10
44/44 [=====] - 518s 12s/step - loss: 0.1160 - accuracy: 0.9566 - val_loss: 0.0223 - val_accuracy: 0.9920
Epoch 5/10
44/44 [=====] - 520s 12s/step - loss: 0.0798 - accuracy: 0.9726 - val_loss: 0.0086 - val_accuracy: 0.9980
Epoch 6/10
44/44 [=====] - 522s 12s/step - loss: 0.0528 - accuracy: 0.9827 - val_loss: 0.0040 - val_accuracy: 1.0000
Epoch 7/10
44/44 [=====] - 521s 12s/step - loss: 0.0382 - accuracy: 0.9871 - val_loss: 0.0032 - val_accuracy: 1.0000
Epoch 8/10
44/44 [=====] - 521s 12s/step - loss: 0.0506 - accuracy: 0.9826 - val_loss: 0.0040 - val_accuracy: 0.9990
Epoch 9/10
44/44 [=====] - 522s 12s/step - loss: 0.0337 - accuracy: 0.9885 - val_loss: 0.0066 - val_accuracy: 0.9990
Epoch 10/10
44/44 [=====] - 521s 12s/step - loss: 0.0217 - accuracy: 0.9923 - val_loss: 7.2733e-04 - val_accuracy: 1.0000

```

Figure 5-4: Training output statement of CNN-AlexNet in fold 3

```

Results for fold 4
Found 11107 images belonging to 2 classes.
Found 1997 images belonging to 2 classes.
Epoch 1/10
44/44 [=====] - 521s 12s/step - loss: 0.0261 - accuracy: 0.9909 - val_loss: 5.1772e-04 - val_accuracy: 1.0000
Epoch 2/10
44/44 [=====] - 520s 12s/step - loss: 0.0209 - accuracy: 0.9926 - val_loss: 6.3614e-04 - val_accuracy: 1.0000
Epoch 3/10
44/44 [=====] - 521s 12s/step - loss: 0.0191 - accuracy: 0.9942 - val_loss: 7.1850e-04 - val_accuracy: 1.0000
Epoch 4/10
44/44 [=====] - 518s 12s/step - loss: 0.0180 - accuracy: 0.9938 - val_loss: 4.3212e-04 - val_accuracy: 1.0000
Epoch 5/10
44/44 [=====] - 520s 12s/step - loss: 0.0183 - accuracy: 0.9943 - val_loss: 5.9748e-04 - val_accuracy: 1.0000
Epoch 6/10
44/44 [=====] - 523s 12s/step - loss: 0.0173 - accuracy: 0.9941 - val_loss: 0.0020 - val_accuracy: 0.9995
Epoch 7/10
44/44 [=====] - 519s 12s/step - loss: 0.0200 - accuracy: 0.9927 - val_loss: 9.3277e-04 - val_accuracy: 1.0000
Epoch 8/10
44/44 [=====] - 519s 12s/step - loss: 0.0110 - accuracy: 0.9963 - val_loss: 0.0030 - val_accuracy: 0.9990
Epoch 9/10
44/44 [=====] - 518s 12s/step - loss: 0.0246 - accuracy: 0.9919 - val_loss: 0.0054 - val_accuracy: 0.9985
Epoch 10/10
44/44 [=====] - 519s 12s/step - loss: 0.0256 - accuracy: 0.9918 - val_loss: 0.0015 - val_accuracy: 0.9995

```

Figure 5-5: Training output statement of CNN-AlexNet in fold 4

```

Results for fold 5
Found 11108 images belonging to 2 classes.
Found 1996 images belonging to 2 classes.
Epoch 1/10
44/44 [=====] - 519s 12s/step - loss: 0.0159 - accuracy: 0.9950 - val_loss: 1.5107e-04 - val_accuracy: 1.0000
Epoch 2/10
44/44 [=====] - 520s 12s/step - loss: 0.0170 - accuracy: 0.9943 - val_loss: 2.3462e-04 - val_accuracy: 1.0000
Epoch 3/10
44/44 [=====] - 521s 12s/step - loss: 0.1884 - accuracy: 0.9294 - val_loss: 0.0828 - val_accuracy: 0.9664
Epoch 4/10
44/44 [=====] - 519s 12s/step - loss: 0.1299 - accuracy: 0.9515 - val_loss: 0.0138 - val_accuracy: 0.9970
Epoch 5/10
44/44 [=====] - 522s 12s/step - loss: 0.0608 - accuracy: 0.9788 - val_loss: 0.0076 - val_accuracy: 0.9980
Epoch 6/10
44/44 [=====] - 538s 12s/step - loss: 0.0411 - accuracy: 0.9851 - val_loss: 0.0034 - val_accuracy: 0.9990
Epoch 7/10
44/44 [=====] - 521s 12s/step - loss: 0.0447 - accuracy: 0.9846 - val_loss: 0.0044 - val_accuracy: 0.9990
Epoch 8/10
44/44 [=====] - 521s 12s/step - loss: 0.0390 - accuracy: 0.9862 - val_loss: 0.0020 - val_accuracy: 0.9995
Epoch 9/10
44/44 [=====] - 521s 12s/step - loss: 0.0270 - accuracy: 0.9906 - val_loss: 0.0024 - val_accuracy: 0.9995
Epoch 10/10
44/44 [=====] - 521s 12s/step - loss: 0.0172 - accuracy: 0.9936 - val_loss: 0.0018 - val_accuracy: 0.9995

```

Figure 5-6: Training output statement of CNN-AlexNet in fold 5

```

Results for fold 1
Found 11107 images belonging to 2 classes.
Found 1997 images belonging to 2 classes.
Epoch 1/10
44/44 [=====] - 243s 5s/step - loss: 0.6825 - accuracy: 0.6540 - val_loss: 0.4883 - val_accuracy: 0.7551
Epoch 2/10
44/44 [=====] - 122s 3s/step - loss: 0.4907 - accuracy: 0.7577 - val_loss: 0.4211 - val_accuracy: 0.8167
Epoch 3/10
44/44 [=====] - 120s 3s/step - loss: 0.4345 - accuracy: 0.7984 - val_loss: 0.3881 - val_accuracy: 0.8262
Epoch 4/10
44/44 [=====] - 119s 3s/step - loss: 0.4085 - accuracy: 0.8052 - val_loss: 0.3604 - val_accuracy: 0.8418
Epoch 5/10
44/44 [=====] - 122s 3s/step - loss: 0.3990 - accuracy: 0.8137 - val_loss: 0.3462 - val_accuracy: 0.8568
Epoch 6/10
44/44 [=====] - 119s 3s/step - loss: 0.3745 - accuracy: 0.8311 - val_loss: 0.3207 - val_accuracy: 0.8648
Epoch 7/10
44/44 [=====] - 115s 3s/step - loss: 0.3580 - accuracy: 0.8359 - val_loss: 0.3106 - val_accuracy: 0.8738
Epoch 8/10
44/44 [=====] - 115s 3s/step - loss: 0.3396 - accuracy: 0.8505 - val_loss: 0.3001 - val_accuracy: 0.8738
Epoch 9/10
44/44 [=====] - 117s 3s/step - loss: 0.3259 - accuracy: 0.8563 - val_loss: 0.2769 - val_accuracy: 0.8888
Epoch 10/10
44/44 [=====] - 115s 3s/step - loss: 0.3159 - accuracy: 0.8597 - val_loss: 0.2874 - val_accuracy: 0.8783

```

Figure 5-7: Training output statement of TL-Inception-V3 in fold 1

```

Results for fold 2
Found 11107 images belonging to 2 classes.
Found 1997 images belonging to 2 classes.
Epoch 1/10
44/44 [=====] - 115s 3s/step - loss: 0.3152 - accuracy: 0.8596 - val_loss: 0.2341 - val_accuracy: 0.9164
Epoch 2/10
44/44 [=====] - 117s 3s/step - loss: 0.2977 - accuracy: 0.8685 - val_loss: 0.2447 - val_accuracy: 0.9094
Epoch 3/10
44/44 [=====] - 118s 3s/step - loss: 0.2822 - accuracy: 0.8808 - val_loss: 0.2231 - val_accuracy: 0.9154
Epoch 4/10
44/44 [=====] - 118s 3s/step - loss: 0.2754 - accuracy: 0.8852 - val_loss: 0.2116 - val_accuracy: 0.9169
Epoch 5/10
44/44 [=====] - 117s 3s/step - loss: 0.2604 - accuracy: 0.8921 - val_loss: 0.1904 - val_accuracy: 0.9329
Epoch 6/10
44/44 [=====] - 119s 3s/step - loss: 0.2577 - accuracy: 0.8917 - val_loss: 0.2136 - val_accuracy: 0.9199
Epoch 7/10
44/44 [=====] - 119s 3s/step - loss: 0.2550 - accuracy: 0.8921 - val_loss: 0.1893 - val_accuracy: 0.9284
Epoch 8/10
44/44 [=====] - 119s 3s/step - loss: 0.2531 - accuracy: 0.8944 - val_loss: 0.1788 - val_accuracy: 0.9339
Epoch 9/10
44/44 [=====] - 118s 3s/step - loss: 0.2426 - accuracy: 0.8993 - val_loss: 0.1786 - val_accuracy: 0.9359
Epoch 10/10
44/44 [=====] - 118s 3s/step - loss: 0.2307 - accuracy: 0.9029 - val_loss: 0.1686 - val_accuracy: 0.9369

```

Figure 5-8: Training output statement of TL-Inception-V3 in fold 2

```

Results for fold 3
Found 11107 images belonging to 2 classes.
Found 1997 images belonging to 2 classes.
Epoch 1/10
44/44 [=====] - 116s 3s/step - loss: 0.2395 - accuracy: 0.8993 - val_loss: 0.1629 - val_accuracy: 0.9424
Epoch 2/10
44/44 [=====] - 118s 3s/step - loss: 0.2330 - accuracy: 0.9009 - val_loss: 0.1671 - val_accuracy: 0.9404
Epoch 3/10
44/44 [=====] - 118s 3s/step - loss: 0.2191 - accuracy: 0.9078 - val_loss: 0.1566 - val_accuracy: 0.9409
Epoch 4/10
44/44 [=====] - 116s 3s/step - loss: 0.2085 - accuracy: 0.9158 - val_loss: 0.1401 - val_accuracy: 0.9509
Epoch 5/10
44/44 [=====] - 120s 3s/step - loss: 0.2115 - accuracy: 0.9121 - val_loss: 0.1509 - val_accuracy: 0.9444
Epoch 6/10
44/44 [=====] - 116s 3s/step - loss: 0.2060 - accuracy: 0.9184 - val_loss: 0.1447 - val_accuracy: 0.9479
Epoch 7/10
44/44 [=====] - 118s 3s/step - loss: 0.2006 - accuracy: 0.9182 - val_loss: 0.1362 - val_accuracy: 0.9499
Epoch 8/10
44/44 [=====] - 122s 3s/step - loss: 0.1886 - accuracy: 0.9248 - val_loss: 0.1442 - val_accuracy: 0.9479
Epoch 9/10
44/44 [=====] - 117s 3s/step - loss: 0.1961 - accuracy: 0.9196 - val_loss: 0.1457 - val_accuracy: 0.9449
Epoch 10/10
44/44 [=====] - 119s 3s/step - loss: 0.1971 - accuracy: 0.9184 - val_loss: 0.1434 - val_accuracy: 0.9449

```

Figure 5-9: Training output statement of TL-Inception-V3 in fold 3

```

Results for fold 4
Found 11107 images belonging to 2 classes.
Found 1997 images belonging to 2 classes.
Epoch 1/10
44/44 [=====] - 117s 3s/step - loss: 0.1809 - accuracy: 0.9288 - val_loss: 0.1225 - val_accuracy: 0.9564
Epoch 2/10
44/44 [=====] - 118s 3s/step - loss: 0.1827 - accuracy: 0.9249 - val_loss: 0.1214 - val_accuracy: 0.9559
Epoch 3/10
44/44 [=====] - 118s 3s/step - loss: 0.1818 - accuracy: 0.9293 - val_loss: 0.1232 - val_accuracy: 0.9544
Epoch 4/10
44/44 [=====] - 118s 3s/step - loss: 0.1758 - accuracy: 0.9282 - val_loss: 0.1266 - val_accuracy: 0.9514
Epoch 5/10
44/44 [=====] - 119s 3s/step - loss: 0.1802 - accuracy: 0.9248 - val_loss: 0.1290 - val_accuracy: 0.9439
Epoch 6/10
44/44 [=====] - 119s 3s/step - loss: 0.1738 - accuracy: 0.9301 - val_loss: 0.1121 - val_accuracy: 0.9579
Epoch 7/10
44/44 [=====] - 119s 3s/step - loss: 0.1782 - accuracy: 0.9272 - val_loss: 0.1156 - val_accuracy: 0.9539
Epoch 8/10
44/44 [=====] - 116s 3s/step - loss: 0.1609 - accuracy: 0.9342 - val_loss: 0.1248 - val_accuracy: 0.9539
Epoch 9/10
44/44 [=====] - 127s 3s/step - loss: 0.1673 - accuracy: 0.9328 - val_loss: 0.1408 - val_accuracy: 0.9464
Epoch 10/10
44/44 [=====] - 129s 3s/step - loss: 0.1739 - accuracy: 0.9277 - val_loss: 0.1101 - val_accuracy: 0.9584

```

Figure 5-10: Training output statement of TL-Inception-V3 in fold 4

```

Results for fold 5
Found 11108 images belonging to 2 classes.
Found 1996 images belonging to 2 classes.
Epoch 1/10
44/44 [=====] - 114s 3s/step - loss: 0.1665 - accuracy: 0.9333 - val_loss: 0.0953 - val_accuracy: 0.9704
Epoch 2/10
44/44 [=====] - 110s 2s/step - loss: 0.1702 - accuracy: 0.9312 - val_loss: 0.0915 - val_accuracy: 0.9724
Epoch 3/10
44/44 [=====] - 111s 3s/step - loss: 0.1626 - accuracy: 0.9368 - val_loss: 0.0999 - val_accuracy: 0.9669
Epoch 4/10
44/44 [=====] - 110s 2s/step - loss: 0.1607 - accuracy: 0.9352 - val_loss: 0.0880 - val_accuracy: 0.9689
Epoch 5/10
44/44 [=====] - 113s 3s/step - loss: 0.1623 - accuracy: 0.9378 - val_loss: 0.0872 - val_accuracy: 0.9699
Epoch 6/10
44/44 [=====] - 117s 3s/step - loss: 0.1545 - accuracy: 0.9373 - val_loss: 0.0875 - val_accuracy: 0.9704
Epoch 7/10
44/44 [=====] - 117s 3s/step - loss: 0.1507 - accuracy: 0.9400 - val_loss: 0.0890 - val_accuracy: 0.9704
Epoch 8/10
44/44 [=====] - 121s 3s/step - loss: 0.1496 - accuracy: 0.9398 - val_loss: 0.0881 - val_accuracy: 0.9729
Epoch 9/10
44/44 [=====] - 118s 3s/step - loss: 0.1503 - accuracy: 0.9400 - val_loss: 0.1023 - val_accuracy: 0.9624
Epoch 10/10
44/44 [=====] - 122s 3s/step - loss: 0.1557 - accuracy: 0.9361 - val_loss: 0.0876 - val_accuracy: 0.9719

```

Figure 5-11: Training output statement of TL-Inception-V3 in fold 5

```

Results for fold 1
Found 11107 images belonging to 2 classes.
Found 1997 images belonging to 2 classes.
Epoch 1/10
44/44 [=====] - 323s 7s/step - loss: 0.6584 - accuracy: 0.7484 - val_loss: 0.4132 - val_accuracy: 0.8072
Epoch 2/10
44/44 [=====] - 235s 5s/step - loss: 0.3995 - accuracy: 0.8289 - val_loss: 0.3344 - val_accuracy: 0.8433
Epoch 3/10
44/44 [=====] - 233s 5s/step - loss: 0.3454 - accuracy: 0.8544 - val_loss: 0.3025 - val_accuracy: 0.8563
Epoch 4/10
44/44 [=====] - 242s 5s/step - loss: 0.2778 - accuracy: 0.8797 - val_loss: 0.2042 - val_accuracy: 0.9334
Epoch 5/10
44/44 [=====] - 247s 6s/step - loss: 0.2502 - accuracy: 0.8944 - val_loss: 0.1861 - val_accuracy: 0.9384
Epoch 6/10
44/44 [=====] - 247s 6s/step - loss: 0.2242 - accuracy: 0.9053 - val_loss: 0.1573 - val_accuracy: 0.9404
Epoch 7/10
44/44 [=====] - 256s 6s/step - loss: 0.2060 - accuracy: 0.9133 - val_loss: 0.1381 - val_accuracy: 0.9544
Epoch 8/10
44/44 [=====] - 247s 6s/step - loss: 0.1777 - accuracy: 0.9279 - val_loss: 0.1166 - val_accuracy: 0.9634
Epoch 9/10
44/44 [=====] - 241s 5s/step - loss: 0.1747 - accuracy: 0.9273 - val_loss: 0.1007 - val_accuracy: 0.9695
Epoch 10/10
44/44 [=====] - 230s 5s/step - loss: 0.1594 - accuracy: 0.9333 - val_loss: 0.0814 - val_accuracy: 0.9745

```

Figure 5-12: Training output statement of TL-DenseNet in fold 1

```

Results for fold 2
Found 11107 images belonging to 2 classes.
Found 1997 images belonging to 2 classes.
Epoch 1/10
44/44 [=====] - 239s 5s/step - loss: 0.1527 - accuracy: 0.9387 - val_loss: 0.0753 - val_accuracy: 0.9710
Epoch 2/10
44/44 [=====] - 255s 6s/step - loss: 0.1459 - accuracy: 0.9443 - val_loss: 0.0704 - val_accuracy: 0.9705
Epoch 3/10
44/44 [=====] - 291s 7s/step - loss: 0.1335 - accuracy: 0.9475 - val_loss: 0.0597 - val_accuracy: 0.9810
Epoch 4/10
44/44 [=====] - 253s 6s/step - loss: 0.1326 - accuracy: 0.9465 - val_loss: 0.0763 - val_accuracy: 0.9710
Epoch 5/10
44/44 [=====] - 257s 6s/step - loss: 0.1285 - accuracy: 0.9491 - val_loss: 0.0568 - val_accuracy: 0.9810
Epoch 6/10
44/44 [=====] - 243s 6s/step - loss: 0.1196 - accuracy: 0.9525 - val_loss: 0.0624 - val_accuracy: 0.9760
Epoch 7/10
44/44 [=====] - 243s 5s/step - loss: 0.1226 - accuracy: 0.9514 - val_loss: 0.0604 - val_accuracy: 0.9775
Epoch 8/10
44/44 [=====] - 250s 6s/step - loss: 0.1172 - accuracy: 0.9533 - val_loss: 0.0460 - val_accuracy: 0.9825
Epoch 9/10
44/44 [=====] - 281s 6s/step - loss: 0.1149 - accuracy: 0.9548 - val_loss: 0.0448 - val_accuracy: 0.9830
Epoch 10/10
44/44 [=====] - 254s 6s/step - loss: 0.1076 - accuracy: 0.9574 - val_loss: 0.0404 - val_accuracy: 0.9830

```

Figure 5-13: Training output statement of TL-DenseNet in fold 2

```

Results for fold 3
Found 11107 images belonging to 2 classes.
Found 1997 images belonging to 2 classes.
Epoch 1/10
44/44 [=====] - 272s 6s/step - loss: 0.1059 - accuracy: 0.9554 - val_loss: 0.0263 - val_accuracy: 0.9910
Epoch 2/10
44/44 [=====] - 272s 6s/step - loss: 0.1055 - accuracy: 0.9597 - val_loss: 0.0209 - val_accuracy: 0.9940
Epoch 3/10
44/44 [=====] - 274s 6s/step - loss: 0.1010 - accuracy: 0.9620 - val_loss: 0.0277 - val_accuracy: 0.9920
Epoch 4/10
44/44 [=====] - 248s 6s/step - loss: 0.0928 - accuracy: 0.9622 - val_loss: 0.0209 - val_accuracy: 0.9945
Epoch 5/10
44/44 [=====] - 244s 6s/step - loss: 0.0976 - accuracy: 0.9628 - val_loss: 0.0186 - val_accuracy: 0.9950
Epoch 6/10
44/44 [=====] - 247s 6s/step - loss: 0.0870 - accuracy: 0.9671 - val_loss: 0.0233 - val_accuracy: 0.9910
Epoch 7/10
44/44 [=====] - 278s 6s/step - loss: 0.0910 - accuracy: 0.9629 - val_loss: 0.0311 - val_accuracy: 0.9885
Epoch 8/10
44/44 [=====] - 265s 6s/step - loss: 0.0843 - accuracy: 0.9689 - val_loss: 0.0415 - val_accuracy: 0.9815
Epoch 9/10
44/44 [=====] - 261s 6s/step - loss: 0.0850 - accuracy: 0.9680 - val_loss: 0.0135 - val_accuracy: 0.9980
Epoch 10/10
44/44 [=====] - 264s 6s/step - loss: 0.0829 - accuracy: 0.9668 - val_loss: 0.0304 - val_accuracy: 0.9880

```

Figure 5-14: Training output statement of TL-DenseNet in fold 3

```

Results for fold 4
Found 11107 images belonging to 2 classes.
Found 1997 images belonging to 2 classes.
Epoch 1/10
44/44 [=====] - 257s 6s/step - loss: 0.0843 - accuracy: 0.9675 - val_loss: 0.0225 - val_accuracy: 0.9920
Epoch 2/10
44/44 [=====] - 254s 6s/step - loss: 0.0773 - accuracy: 0.9704 - val_loss: 0.0188 - val_accuracy: 0.9925
Epoch 3/10
44/44 [=====] - 278s 6s/step - loss: 0.0748 - accuracy: 0.9717 - val_loss: 0.0153 - val_accuracy: 0.9960
Epoch 4/10
44/44 [=====] - 258s 6s/step - loss: 0.0744 - accuracy: 0.9723 - val_loss: 0.0140 - val_accuracy: 0.9970
Epoch 5/10
44/44 [=====] - 240s 6s/step - loss: 0.0783 - accuracy: 0.9694 - val_loss: 0.0184 - val_accuracy: 0.9945
Epoch 6/10
44/44 [=====] - 250s 6s/step - loss: 0.0769 - accuracy: 0.9690 - val_loss: 0.0147 - val_accuracy: 0.9960
Epoch 7/10
44/44 [=====] - 248s 6s/step - loss: 0.0620 - accuracy: 0.9780 - val_loss: 0.0211 - val_accuracy: 0.9920
Epoch 8/10
44/44 [=====] - 254s 6s/step - loss: 0.0651 - accuracy: 0.9748 - val_loss: 0.0151 - val_accuracy: 0.9950
Epoch 9/10
44/44 [=====] - 246s 6s/step - loss: 0.0667 - accuracy: 0.9743 - val_loss: 0.0137 - val_accuracy: 0.9965
Epoch 10/10
44/44 [=====] - 252s 6s/step - loss: 0.0626 - accuracy: 0.9763 - val_loss: 0.0146 - val_accuracy: 0.9955

```

Figure 5-15: Training output statement of TL-DenseNet in fold 4

```

Results for fold 5
Found 11108 images belonging to 2 classes.
Found 1996 images belonging to 2 classes.
Epoch 1/10
44/44 [=====] - 229s 5s/step - loss: 0.0700 - accuracy: 0.9744 - val_loss: 0.0090 - val_accuracy: 0.9980
Epoch 2/10
44/44 [=====] - 228s 5s/step - loss: 0.0672 - accuracy: 0.9741 - val_loss: 0.0137 - val_accuracy: 0.9960
Epoch 3/10
44/44 [=====] - 229s 5s/step - loss: 0.0624 - accuracy: 0.9754 - val_loss: 0.0071 - val_accuracy: 0.9990
Epoch 4/10
44/44 [=====] - 274s 6s/step - loss: 0.0591 - accuracy: 0.9771 - val_loss: 0.0083 - val_accuracy: 0.9975
Epoch 5/10
44/44 [=====] - 266s 6s/step - loss: 0.0601 - accuracy: 0.9762 - val_loss: 0.0069 - val_accuracy: 0.9990
Epoch 6/10
44/44 [=====] - 264s 6s/step - loss: 0.0591 - accuracy: 0.9771 - val_loss: 0.0074 - val_accuracy: 0.9985
Epoch 7/10
44/44 [=====] - 259s 6s/step - loss: 0.0541 - accuracy: 0.9782 - val_loss: 0.0046 - val_accuracy: 1.0000
Epoch 8/10
44/44 [=====] - 253s 6s/step - loss: 0.0585 - accuracy: 0.9779 - val_loss: 0.0060 - val_accuracy: 1.0000
Epoch 9/10
44/44 [=====] - 251s 6s/step - loss: 0.0538 - accuracy: 0.9809 - val_loss: 0.0049 - val_accuracy: 1.0000
Epoch 10/10
44/44 [=====] - 252s 6s/step - loss: 0.0558 - accuracy: 0.9793 - val_loss: 0.0076 - val_accuracy: 0.9985

```

Figure 5-16: Training output statement of TL-DenseNet in fold 5

```

=====TEST RESULTS=====
Found 480 images belonging to 2 classes.
15/15 [=====] - 6s 352ms/step
Accuracy : 0.9041666666666667
Precision : 0.9105820105820105
f1Score : 0.9037908496732026
[[232  8]
 [ 38 202]]

```

Figure 5-17: Testing output statement of CNN-AlexNet

```

=====TEST RESULTS=====
Found 480 images belonging to 2 classes.
15/15 [=====] - 6s 243ms/step
Accuracy : 0.9104166666666667
Precision : 0.9173802503575275
f1Score : 0.9100414489254225
[[203 37]
 [  6 234]]

```

Figure 5-18: Testing output statement of TL-Inception-V3

```

=====TEST RESULTS=====
Found 480 images belonging to 2 classes.
15/15 [=====] - 12s 610ms/step
Accuracy : 0.9145833333333333
Precision : 0.9171980642568879
f1Score : 0.9144492890335987
[[229 11]
 [ 30 210]]

```

Figure 5-19: Testing output statement of TL-DenseNet

APPENDIX B: Computer Programme Listing

Coding for Image Processing and Augmentation.

```
import glob
import cv2
import os
import numpy as np
from matplotlib import pyplot as plt

inputFolder =
"C:/Users/zzxn9/PycharmProjects/dashproject/Cancer/Gans/new/malignant"
# os.mkdir("C:/Users/zzxn9/Documents/Py-DS-ML-Bootcamp-
master/Refactored_Py_DS_ML_Bootcamp-master/Cancer/Process/benign")

i=1
j=200
k=400
l=600

kernel_sharpening = np.array([
    [0, -1, 0],
    [-1, 5, -1],
    [0, -1, 0]
])

kernel2 = np.ones((5, 5), np.float32) / 25
dim = (100, 100)

a=1601
b=3201
c=4801

for filename in os.listdir(inputFolder):
    image = cv2.imread(os.path.join(inputFolder,filename))
    if image is not None:

        conv2d = cv2.filter2D(src=image, ddepth=-1, kernel=kernel2)
        denoise = cv2.fastNlMeansDenoisingColored(image, None, 10, 10, 7, 21)
        gaussian_blur = cv2.GaussianBlur(src=image, ksize=(5, 5),sigmaX = 0,
sigmaY = 0)
        median = cv2.medianBlur(src=image, ksize=5)
        bilateral_filter = cv2.bilateralFilter(src=image, d=9, sigmaColor=75,
sigmaSpace=75)
        output = cv2.filter2D(image, -1, kernel_sharpening)
        cropped_image = image[60:410, 150:500]

        gray = cv2.cvtColor(image, cv2.COLOR_BGR2GRAY)
```

```
resized = cv2.resize(image, dim, interpolation=cv2.INTER_AREA)

    flipVertical = cv2.flip(output, 0)
    flipHorizontal = cv2.flip(output, 1)
    flipBoth = cv2.flip(output, -1)
    # save and display images
    cv2.imwrite(

"C:/Users/zxxn9/PycharmProjects/dashproject/Cancer/Gans/new/malignant/us_m
%02i.jpg" % i,
    gray)
    i += 1
    cv2.imwrite(

"C:/Users/zxxn9/PycharmProjects/dashproject/Cancer/Gans/new/malignant/us_b
%02i.jpg" % i,
    resized)
    i += 1
    cv2.imwrite(

"C:/Users/zxxn9/PycharmProjects/dashproject/Cancer/Gans/benign/us_m%02i.jp
g" % i,
    cropped_image)
    i += 1
    cv2.imwrite(

"C:/Users/zxxn9/PycharmProjects/dashproject/Cancer/Conv2d/benign/us_m%02i
.jpg" % i,
    conv2d)
    i += 1
    cv2.imwrite(

"C:/Users/zxxn9/PycharmProjects/dashproject/Cancer/Gans/new/malignant/us_m
%02i.jpg" % i,
    denoise)
    i += 1
    cv2.imwrite(

"C:/Users/zxxn9/PycharmProjects/dashproject/Cancer/Bilateral/gans/us_m%02i.j
pg" % i,
    bilateral_filter)
    i += 1
    cv2.imwrite(

"C:/Users/zxxn9/PycharmProjects/dashproject/Cancer/Median/gans/us_m%02i.jp
g" % i,
    median)
    i += 1
```

```
cv2.imwrite(  
  
"C:/Users/zxxn9/PycharmProjects/dashproject/Cancer/Gaussian/gans/us_m%02i.j  
pg" % i,  
    gaussian_blur)  
    i += 1  
    cv2.imwrite(  
  
"C:/Users/zxxn9/PycharmProjects/dashproject/Cancer/Sharp/benign/us_m%02i.j  
pg" % i,  
    output)  
    i += 1  
    cv2.imwrite(  
  
"C:/Users/zxxn9/PycharmProjects/dashproject/Cancer/Process/gans/malignant/us  
_b%02i.jpg" % j,  
    flipVertical)  
    j += 1  
    cv2.imwrite(  
  
"C:/Users/zxxn9/PycharmProjects/dashproject/Cancer/Process/gans/malignant/us  
_b%02i.jpg" % k,  
    flipHorizontal)  
    k += 1  
    cv2.imwrite(  
  
"C:/Users/zxxn9/PycharmProjects/dashproject/Cancer/Process/gans/malignant/us  
_b%02i.jpg" % l,  
    flipBoth)  
    l += 1  
  
#press esc to exit the program  
cv2.waitKey(30)  
#close all the opened windows  
cv2.destroyAllWindows()
```

Coding for Rotating Images

```

img = cv2.imread(os.path.join(inputFolder,filename))
(h, w) = img.shape[:2]
# calculate the center of the image
center = (w / 2, h / 2)
angle45 = 45
angle125 = 125
angle315 = 315
scale = 1.0
# Perform the counter clockwise rotation holding at the center
# 45 degrees
M = cv2.getRotationMatrix2D(center, angle45, scale)
rotated45 = cv2.warpAffine(img, M, (h, w))
cv2.imwrite(

"C:/Users/zxxn9/PycharmProjects/dashproject/Cancer/Report/us_m%02i.jpg" %
a,
    rotated45)
a += 1
# 125 degrees
M = cv2.getRotationMatrix2D(center, angle125, scale)
rotated125 = cv2.warpAffine(img, M, (w, h))
cv2.imwrite(

"C:/Users/zxxn9/PycharmProjects/dashproject/Cancer/Report/us_m%02i.jpg" %
b,
    rotated125)
b += 1
# 315 degrees
M = cv2.getRotationMatrix2D(center, angle315, scale)
rotated315 = cv2.warpAffine(img, M, (h, w))
cv2.imwrite(

"C:/Users/zxxn9/PycharmProjects/dashproject/Cancer/Report/us_m%02i.jpg" %
c,
    rotated315)
c += 1

```

Coding for DCGANs model

```

import os
import numpy as np
import cv2
from glob import glob
from matplotlib import pyplot
from sklearn.utils import shuffle
import tensorflow as tf
from tensorflow.keras.layers import *
from tensorflow.keras.models import Model
from tensorflow.keras.optimizers import Adam
from skimage import color

IMG_H = 80
IMG_W = 80
IMG_C = 3
w_init = tf.keras.initializers.RandomNormal(mean=0.0, stddev=0.02)
def load_image(image_path):
    img = tf.io.read_file(image_path)
    img = tf.io.decode_jpeg(img)
    img = tf.image.resize_with_crop_or_pad(img, IMG_H, IMG_W)
    img = tf.cast(img, tf.float32)
    img = (img - 127.5) / 127.5
    return img

def tf_dataset(images_path, batch_size):
    dataset = tf.data.Dataset.from_tensor_slices(images_path)
    dataset = dataset.shuffle(buffer_size=10240)
    dataset = dataset.map(load_image,
num_parallel_calls=tf.data.experimental.AUTOTUNE)
    dataset = dataset.batch(batch_size)
    dataset = dataset.prefetch(buffer_size=tf.data.experimental.AUTOTUNE)
    return dataset

def deconv_block(inputs, num_filters, kernel_size, strides, bn=True):
    x = Conv2DTranspose(
        filters=num_filters,
        kernel_size=kernel_size,
        kernel_initializer=w_init,
        padding="same",
        strides=strides,
        use_bias=False
    )(inputs)

    if bn:
        x = BatchNormalization()(x)
        x = LeakyReLU(alpha=0.2)(x)
    return x

```

```

def conv_block(inputs, num_filters, kernel_size, padding="same", strides=2,
activation=True):
    x = Conv2D(
        filters=num_filters,
        kernel_size=kernel_size,
        kernel_initializer=w_init,
        padding=padding,
        strides=strides,
    )(inputs)

    if activation:
        x = LeakyReLU(alpha=0.2)(x)
        x = Dropout(0.3)(x)
    return x

def build_generator(latent_dim):
    f = [2**i for i in range(5)][::-1]
    filters = 32
    output_strides = 16
    h_output = IMG_H // output_strides
    w_output = IMG_W // output_strides

    noise = Input(shape=(latent_dim,), name="generator_noise_input")

    x = Dense(f[0] * filters * h_output * w_output, use_bias=False)(noise)
    x = BatchNormalization()(x)
    x = LeakyReLU(alpha=0.2)(x)
    x = Reshape((h_output, w_output, 16 * filters))(x)

    for i in range(1, 5):
        x = deconv_block(x,
            num_filters=f[i] * filters,
            kernel_size=5,
            strides=2,
            bn=True
        )

    x = conv_block(x,
        num_filters=3,
        kernel_size=5,
        strides=1,
        activation=False
    )
    fake_output = Activation("tanh")(x)

    return Model(noise, fake_output, name="generator")

```

```

def build_discriminator():
    f = [2**i for i in range(4)]
    image_input = Input(shape=(IMG_H, IMG_W, IMG_C))
    x = image_input
    filters = 64
    output_strides = 16
    h_output = IMG_H // output_strides
    w_output = IMG_W // output_strides

    for i in range(0, 4):
        x = conv_block(x, num_filters=f[i] * filters, kernel_size=5, strides=2)

    x = Flatten()(x)
    x = Dense(1)(x)

    return Model(image_input, x, name="discriminator")

class GAN(Model):
    def __init__(self, discriminator, generator, latent_dim):
        super(GAN, self).__init__()
        self.discriminator = discriminator
        self.generator = generator
        self.latent_dim = latent_dim

    def compile(self, d_optimizer, g_optimizer, loss_fn):
        super(GAN, self).compile()
        self.d_optimizer = d_optimizer
        self.g_optimizer = g_optimizer
        self.loss_fn = loss_fn

    def train_step(self, real_images):
        batch_size = tf.shape(real_images)[0]

        for _ in range(2):
            ## Train the discriminator
            random_latent_vectors = tf.random.normal(shape=(batch_size,
self.latent_dim))
            generated_images = self.generator(random_latent_vectors)
            generated_labels = tf.zeros((batch_size, 1))

            with tf.GradientTape() as ftape:
                predictions = self.discriminator(generated_images)
                d1_loss = self.loss_fn(generated_labels, predictions)
                grads = ftape.gradient(d1_loss, self.discriminator.trainable_weights)
                self.d_optimizer.apply_gradients(zip(grads,
self.discriminator.trainable_weights))

```

```

## Train the discriminator
    labels = tf.ones((batch_size, 1))

    with tf.GradientTape() as rtape:
        predictions = self.discriminator(real_images)
        d2_loss = self.loss_fn(labels, predictions)
        grads = rtape.gradient(d2_loss, self.discriminator.trainable_weights)
        self.d_optimizer.apply_gradients(zip(grads,
self.discriminator.trainable_weights))

    ## Train the generator
    random_latent_vectors = tf.random.normal(shape=(batch_size,
self.latent_dim))
    misleading_labels = tf.ones((batch_size, 1))

    with tf.GradientTape() as gtape:
        predictions = self.discriminator(self.generator(random_latent_vectors))
        g_loss = self.loss_fn(misleading_labels, predictions)
        grads = gtape.gradient(g_loss, self.generator.trainable_weights)
        self.g_optimizer.apply_gradients(zip(grads,
self.generator.trainable_weights))

    return {"d1_loss": d1_loss, "d2_loss": d2_loss, "g_loss": g_loss}

def save_plot(examples, epoch, n):
    examples = (examples + 1) / 2.0
    for i in range(n * n):
        pyplot.subplot(n, n, i+1)
        pyplot.axis("off")
        pyplot.imshow(examples[i]) ## pyplot.imshow(np.squeeze(examples[i],
axis=-1))
    filename = f"samples_new1/generated_plot_epoch-{{epoch+1}}.png"
    pyplot.savefig(filename)
    pyplot.close()

```

Coding for training the DCGANs model

```

if __name__ == "__main__":
    ## Hyperparameters
    batch_size = 256
    latent_dim = 256
    num_epochs = 800
    images_path = glob("data/*")

    d_model = build_discriminator()
    g_model = build_generator(latent_dim)

    d_model.load_weights("saved_model/d_model.h5")
    g_model.load_weights("saved_model/g_model.h5")

    d_model.summary()
    g_model.summary()

    gan = GAN(d_model, g_model, latent_dim)

    bce_loss_fn = tf.keras.losses.BinaryCrossentropy(from_logits=True,
label_smoothing=0.1)
    d_optimizer = tf.keras.optimizers.Adam(learning_rate=0.0002, beta_1=0.5)
    g_optimizer = tf.keras.optimizers.Adam(learning_rate=0.0002, beta_1=0.5)
    gan.compile(d_optimizer, g_optimizer, bce_loss_fn)

    images_dataset = tf_dataset(images_path, batch_size)

    for epoch in range(num_epochs):
        gan.fit(images_dataset, epochs=1)
        g_model.save("saved_model/g_model.h5")
        d_model.save("saved_model/d_model.h5")

        n_samples = 25
        noise = np.random.normal(size=(n_samples, latent_dim))
        examples = g_model.predict(noise)
        # save_plot(examples, epoch, int(np.sqrt(n_samples)))
        if epoch % 2 != 0:
            save_plot(examples, epoch, int(np.sqrt(n_samples)))
            g_model.save(f"saved_model_acc/g_model-{{epoch+1}}.h5")
            d_model.save(f"saved_model_acc/d_model-{{epoch+1}}.h5")

```

Coding for generating synthesized images from the DCGANs model

```

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

def save_plot(examples, n):
    examples = (examples + 1) / 2.0
    for i in range(n):
        pyplot.imshow(examples[i])
        pyplot.axis("off")
        filename = f"malignant{+1}.png"
        pyplot.savefig(filename, bbox_inches='tight', pad_inches = 0)
        pyplot.close()

if __name__ == "__main__":
    model = load_model("C:/Users/zzxn9/PycharmProjects/dashproject/DCGAN-
on-Breast-tumor/saved_model/g_model.h5", compile=False)

    n = 25
    latent_dim = 256
    latent_points = np.random.normal(size=(n, latent_dim))
    examples = model.predict(latent_points)
    save_plot(examples, n)

```

Coding for splitting datasets into train, validation and test sets

```

import splitfolders

splitfolders.ratio("C:/Users/zzxn9/PycharmProjects/dashproject/Cancer/Rotation/t
est-set",
                 output="C:/Users/zzxn9/PycharmProjects/dashproject/Cancer/Cheat",
                 seed=42,
                 ratio=(.7, .2, .1),
                 group_prefix=None,
                 move=False)

```

Coding for the proposed CNN network

```

import numpy as np
from sklearn.metrics import accuracy_score, f1_score, precision_score,
confusion_matrix
from sklearn.model_selection import StratifiedKFold
from PIL import Image
import random
import tensorflow as tf
from tensorflow import keras
from tensorflow.keras.applications import DenseNet121
from tensorflow.keras.applications.resnet50 import ResNet50
from tensorflow.keras.applications.inception_v3 import InceptionV3
from tensorflow.keras import layers, Model
from tensorflow.keras.models import Sequential
from tensorflow.keras.layers import Dense
from keras.preprocessing.image import ImageDataGenerator
from tensorflow.keras.layers import Dropout
from tensorflow.keras.layers import Flatten
from tensorflow.keras.layers import Conv2D
from tensorflow.keras.layers import MaxPooling2D
from tensorflow.compat.v1 import ConfigProto
from tensorflow.compat.v1 import InteractiveSession
from tensorflow.keras.layers import
Dense,GlobalAveragePooling2D,Convolution2D,BatchNormalization
import matplotlib.pyplot as plt
from sklearn import metrics
import warnings
import os
import shutil
from PIL import ImageFile
import zipfile
import matplotlib.pyplot as plt

warnings.simplefilter('error', Image.DecompressionBombWarning)
ImageFile.LOAD_TRUNCATED_IMAGES = True
from PIL import Image
Image.MAX_IMAGE_PIXELS = 10000
config = ConfigProto()
config.gpu_options.allow_growth = True
session = InteractiveSession(config=config)

root_path='C:/Users/zxxn9/PycharmProjects/dashproject/Cancer/'
datasetFolderName=root_path+'Training'
MODEL_FILENAME=root_path+"model_cv.h5"
sourceFiles=[]
classLabels=['benign', 'malignant']
X=[]
Y=[]

```

```

Image.MAX_IMAGE_PIXELS = None
img_rows, img_cols = 100, 100 # input image dimensions
train_path=datasetFolderName+'/train/'
validation_path=datasetFolderName+'/val/'
test_path=datasetFolderName+'/test/'

def transferBetweenFolders(source, dest, splitRate):
    global sourceFiles
    sourceFiles=os.listdir(source)
    if(len(sourceFiles)!=0):
        transferFileNumbers=int(len(sourceFiles)*splitRate)
        transferIndex=random.sample(range(0, len(sourceFiles)),
transferFileNumbers)
        for eachIndex in transferIndex:
            shutil.move(source+str(sourceFiles[eachIndex]),
dest+str(sourceFiles[eachIndex]))
    else:
        print("No file moved. Source empty!")

def transferAllClassBetweenFolders(source, dest, splitRate):
    for label in classLabels:
        transferBetweenFolders(datasetFolderName + '/' + source + '/' + label + '/',
datasetFolderName + '/' + dest + '/' + label + '/',
splitRate)

transferAllClassBetweenFolders('test', 'train', 1.0)
transferAllClassBetweenFolders('train', 'test', 0.20)

def prepareNameWithLabels(folderName):
    sourceFiles = os.listdir(datasetFolderName + '/train/' + folderName)
    for val in sourceFiles:
        X.append(val)
        for i in range(len(classLabels)):
            if (folderName == classLabels[i]):
                Y.append(i)
# Organize file names and class labels in X and Y variables
for i in range(len(classLabels)):
    prepareNameWithLabels(classLabels[i])

X = np.asarray(X)
Y = np.asarray(Y)

batch_size = 256
epoch = 10
activationFunction = 'elu'

```

Coding for CNN-AlexNet

```

def getModel():

    # AlexNet
    model = Sequential()
    model.add(Conv2D(64, (3, 3), padding='same', activation=activationFunction,
input_shape=(img_rows, img_cols, 3)))
    model.add(Conv2D(64, (3, 3), activation=activationFunction))
    model.add(MaxPooling2D(pool_size=(2, 2)))
    model.add(Dropout(0.25))
    model.add(Conv2D(32, (3, 3), padding='same', activation=activationFunction))
    model.add(Conv2D(32, (3, 3), activation=activationFunction))
    model.add(MaxPooling2D(pool_size=(2, 2)))
    model.add(Dropout(0.25))
    model.add(Conv2D(16, (3, 3), padding='same', activation=activationFunction))
    model.add(Conv2D(16, (3, 3), activation=activationFunction))
    model.add(MaxPooling2D(pool_size=(2, 2)))
    model.add(Dropout(0.25))
    model.add(Flatten())
    model.add(Dense(64, activation=activationFunction))
    model.add(Dropout(0.1))
    model.add(Dense(32, activation=activationFunction))
    model.add(Dropout(0.1))
    model.add(Dense(16, activation=activationFunction))
    model.add(Dropout(0.1))
    model.add(Dense(len(classLabels), activation='softmax'))

```

Coding for TL-Inception-V3

```

# InceptionV3
model_d = InceptionV3(input_shape=(100, 100,
3),include_top=False,weights='imagenet')
model_d.trainable = False
model = Sequential()
model.add(model_d)
model.add(GlobalAveragePooling2D())
model.add(Dropout(0.2))
model.add(Flatten())
model.add(Dense(128, activation='relu', input_dim=(100, 100, 3)))
model.add(Dense(len(classLabels), activation='softmax'))

```

Coding for TL-DenseNet

```
# DenseNet
model_d = DenseNet121(weights='imagenet', include_top=False,
input_shape=(100, 100, 3))
model_d.trainable = False

model = Sequential()
model.add(model_d)
model.add(GlobalAveragePooling2D())
model.add(BatchNormalization())
model.add(Dropout(0.5))
model.add(Dense(1024, activation='relu'))
model.add(Dense(512, activation='relu'))
model.add(BatchNormalization())
model.add(Dropout(0.5))
model.add(Dense(len(classLabels), activation='softmax'))

model.compile(optimizer='adam', loss='binary_crossentropy',
metrics=['accuracy'])

model.summary()

return model

model=getModel()
```

Coding for training and validate the proposed models in Stratified K-Fold

```

skf = StratifiedKFold(n_splits=5, shuffle=True)
skf.get_n_splits(X, Y)
foldNum=0
for train_index, val_index in skf.split(X, Y):

    transferAllClassBetweenFolders('val', 'train', 1.0)
    foldNum+=1
    print("Results for fold",foldNum)
    X_train, X_val = X[train_index], X[val_index]
    Y_train, Y_val = Y[train_index], Y[val_index]
    for eachIndex in range(len(X_val)):
        classLabel=""
        for i in range(len(classLabels)):
            if(Y_val[eachIndex]==i):
                classLabel=classLabels[i]

        shutil.move(datasetFolderName+'/train/'+classLabel+''+X_val[eachIndex],
                    datasetFolderName+'/val/'+classLabel+''+X_val[eachIndex])

train_datagen = ImageDataGenerator(
    rescale=1./255,
    zoom_range=0.20,
    fill_mode="nearest")
validation_datagen = ImageDataGenerator(rescale=1./255)
test_datagen = ImageDataGenerator(rescale=1./255)

#Start ImageClassification Model
train_generator = train_datagen.flow_from_directory(
    train_path,
    target_size=(img_rows, img_cols),
    batch_size=batch_size,
    class_mode='categorical',
    subset='training')

validation_generator = validation_datagen.flow_from_directory(
    validation_path,
    target_size=(img_rows, img_cols),
    batch_size=batch_size,
    class_mode='categorical', # only data, no labels
    shuffle=False)

# fit model
history=model.fit(train_generator,epochs=epoch, validation_data =
validation_generator)

```

```

print("=====TEST RESULTS=====")
test_generator = test_datagen.flow_from_directory(
    test_path,
    target_size=(img_rows, img_cols),
    batch_size=batch_size,
    class_mode=None,
    shuffle=False)

predictions = model.predict_generator(test_generator, verbose=1)
yPredictions = np.argmax(predictions, axis=1)
true_classes = test_generator.classes

confusion_matrix = metrics.confusion_matrix(true_classes, yPredictions)
cm_display =
metrics.ConfusionMatrixDisplay(confusion_matrix=confusion_matrix,
display_labels=[False, True])

cm_display.plot()
plt.title(f"Confusion Matrix - fold {foldNum}")
plt.savefig(f"Confusion Matrix - {foldNum}.png", bbox_inches='tight')
plt.close()

# Plot the training and validation accuracies for each epoch

acc = history.history['accuracy']
val_acc = history.history['val_accuracy']
loss = history.history['loss']
val_loss = history.history['val_loss']

epochs = range(len(acc))

plt.plot(epochs, acc, 'r', label='Training accuracy')
plt.plot(epochs, val_acc, 'b', label='Validation accuracy')
plt.title(f"Training and validation accuracy- fold {foldNum}")
plt.xlabel('Epoch')
plt.ylabel('Accuracy')
plt.legend(loc=0)
plt.savefig(f"training-accuracy{foldNum}.png", bbox_inches='tight')
plt.close()

plt.plot(epochs, loss, 'r', label='Training loss')
plt.plot(epochs, val_loss, 'b', label='Validation loss')
plt.title(f"Training and validation loss- fold {foldNum}")
plt.xlabel('Epoch')
plt.ylabel('Loss')
plt.legend(loc=0)
plt.savefig(f"training-loss{foldNum}.png", bbox_inches='tight')
plt.close()

model.save(MODEL_FILENAME)

```