Breast Cancer Detection in Mammograms using Deep Learning Techniques

Adam Jaamour

Supervised by Dr David Harris-Birtill & Lewis McMillan

Master's of Science in Artificial Intelligence
University of St Andrews - School of Computer Science
August 14, 2020

Breast Cancer Detection in Mammograms using Deep Learning Techniques

Submitted by: Adam Jaamour

I declare that the material submitted for assessment is my own work except where credit is explicitly given to others by citation or acknowledgement. This work was performed during the current academic year except where otherwise stated.

The main text of this project report is 14,345 words long (calculated by Overleaf word counter), including project specification and plan.

Signed:

Abstract

The objective of this dissertation is to explore various deep learning techniques that can be used to implement a system which learns how to detect instances of breast cancer in mammograms. Nowadays, breast cancer claims 11,400 lives on average every year in the UK, making it one of the deadliest diseases. Mammography is the gold standard for detecting early signs of breast cancer, which can help cure the disease during its early stages. However, incorrect mammography diagnoses are common and may harm patients through unnecessary treatments and operations (or a lack of treatments). Therefore, systems that can learn to detect breast cancer on their own could help reduce the number of incorrect interpretations and missed cases.

Convolution Neural Networks (CNNs) are used as part of a deep learning pipeline initially developed in a group and further extended individually. A bag-of-tricks approach is followed to analyse the effects on performance and efficiency using diverse deep learning techniques such as different architectures (VGG19, ResNet50, InceptionV3, DenseNet121, MobileNetV2), class weights, input sizes, amounts of transfer learning, and types of mammograms.

Ultimately, 67.08% accuracy is achieved on the CBIS-DDSM dataset by transfer learning pre-trained ImagetNet weights to a MobileNetV2 architecture and pre-trained weights from a binary version of the mini-MIAS dataset to the fully connected layers of the model. Furthermore, using class weights to fight the problem of imbalanced datasets and splitting CBIS-DDSM samples between masses and calcifications also increases the overall accuracy. Other techniques tested such as data augmentation and larger image sizes do not yield increased accuracies, while the mini-MIAS dataset proves to be too small for any meaningful results using deep learning techniques. These results are compared with other papers using the CBIS-DDSM and mini-MIAS datasets, and with the baseline set from deep learning pipeline developed as a group.

Keywords

Deep Learning; Convolution Neural Networks; Transfer Learning; Breast Cancer Detection; Mammogram Classification; CBIS-DDSM; mini-MIAS

Contents

List of Figures **vi**

List of Tables **ix**

List of Acronyms **x**

1 Introduction **1**
1.1 Motivation 1
1.2 Problem Description 2
1.3 Objectives 3
1.4 Report Structure 3

2 Context Survey **5**
2.1 Breast Cancer Detection 5
2.1.1 Medical imagery screening tests & biopsies 5
2.1.2 Early Breast Cancer Detection Systems 6
2.1.3 Towards Supervised Machine Learning-based Systems . 6
2.2 Machine Learning Tasks & Algorithms 7
2.2.1 Machine Learning Applications to Breast Cancer Detection 7
Types of machine learning algorithms 7
Types of machine learning tasks 8
2.2.2 Comparison of BCD Supervised Learning Algorithms . 10
k-Nearest Neighbours 10
Naive Bayes 11
Decision Trees 12
Support Vector Machines 13
Artificial Neural Networks 15
Supervised machine learning algorithms comparison . . 17
2.3 CNNs & Deep Learning techniques 17
2.3.1 Convolution Neural Networks 17
Motivation for CNNs over traditional neural networks . 17
CNN structure 18

CNN Architectures 20
2.3.2 Deep Learning Applications in Breast Cancer Detection 21
Main challenges 21
Transfer learning 21
Regularisation techniques 22
Technological advances 24
2.4 Summary 24

3 Ethics & Datasets 26
3.1 Ethical Considerations 26
3.2 Datasets Description 26
3.2.1 DDSM 26
3.2.2 CBIS-DDSM 27
3.2.3 mini-MIAS 28

4 Design 29
4.1 Datasets Decision 29
4.2 Deep Learning Pipeline Design Analysis 30
4.2.1 Data Pre-Processing 30
Dataset balance 30
Dataset split 31
Data loading 32
Data normalisation 32
Label encoding 33
4.2.2 Model Training 34
CNN model 34
Data fitting 34
4.2.3 Result Visualisation 37
Overall accuracy 37
Precision & recall 38
F1 score 38
Confusion matrix 38
4.3 General Design Decisions 38
4.3.1 Programming Language 38
4.3.2 Deep Learning Framework 39
4.3.3 Interface 39
4.4 Design Decisions Summary 39

5 Implementation 41
5.1 Code Design 41
5.2 General 42
5.2.1 Command-Line Interface 42
5.2.2 Results reproducibility 42
5.3 Data Pre-Processing 43

5.3.1 Initial Dataset Processing . . . 43
5.3.2 Data Loading . . . 43
5.3.3 Data Processing . . . 43
5.3.4 Dataset Splits . . . 44
5.3.5 Data Augmentation & Class Balance . . . 44
5.4 Model Training . . . 45
5.4.1 Sequential Model . . . 45
5.4.2 Training Steps . . . 46
5.4.3 Model & Weights Saving . . . 47
5.5 Predictions & Results visualisation . . . 47
5.6 Pipeline Flowchart . . . 47

6 Results & Evaluation **49**
6.1 Test Data . . . 49
6.2 Model Used . . . 49
6.3 Baseline Results . . . 50
6.4 Base CNN Architectures . . . 50
6.5 Class Imbalance . . . 51
6.5.1 Data Augmentation . . . 51
6.5.2 Class Weights . . . 53
6.6 Input Image Size . . . 55
6.7 Varying Amounts of Transfer Learning . . . 57
6.8 Mammogram Types . . . 59
6.9 Results Summary . . . 61

7 Conclusions **63**
7.1 Achievements . . . 63
7.2 Code Availability . . . 63
7.3 Limitations . . . 64
7.4 Future Work . . . 65
7.5 Reflections . . . 65

Bibliography **66**

A Ethical Application Approval Letter **74**

B Languages & Frameworks Comparison **76**
B.1 Programming Languages . . . 76
B.2 Deep Learning Frameworks . . . 76

C Usage Instructions **78**
C.1 Installation Instructions . . . 78
C.2 Individual Code Instructions . . . 78
C.3 Common Pipeline Code Instructions . . . 79

C.4 Dataset Installation Instructions 80
C.4.1 mini-MIAS dataset 80
C.4.2 CBIS-DDSM dataset 80

D Remote Work Environment **82**
D.1 Coding environment 82
D.2 Code collaboration 83
D.3 Supervisor meetings 83

E Team Meeting Summaries **84**

F Coding Project Structure **94**

List of Figures

1.1 Example of three types of mammograms, including normal, benign and malignant cases. Figures extracted from the mini-MIAS dataset (Suckling, 1994). Created using draw.io. 2

2.1 Timeline of the evolution of breast cancer detection (BCD) systems synthesising the information described in Sections 2.1.1 and 2.1.2. 7
2.2 Example of a breast mammogram classification, showing benign (left) and malignant (right) mammograms. Images retrieved from the mini-MIAS dataset (Suckling, 1994). 9
2.3 Example of a breast mammogram segmentation, showing the original mammogram (left) and the segmented image (right), depicting large masses. Images retrieved from Punithaet al. (2018). 10
2.4 Example of a kNN classifier distinguishing between benign (blue square) and malignant (red triangle) tumours for a test data sample (green circle) using $k = 3$. The test sample is classified as malignant as there are two red triangles and one blue square amongst the three neighbours. Figure retrieved from T. Srivastava (`https://tinyurl.com/y3jqco49`). 11
2.5 Example of a decision tree classifier distinguishing between benign and malignant tumours based on three extracted features from a dataset of mammograms: the size of the bare nuclei, the thickness of the clump and the uniformity of the cell size. Figure created by Yue et al. (2018). 13
2.6 Example of a SVM classifier's maximum margin hyperplane found to separate benign and malignant tumours based on two extracted features from a dataset of mammograms: the size of the bare nuclei and the uniformity of the cell size. Figure created by Yue et al. (2018). 14
2.7 Example of an ANN classifier distinguishing between benign and malignant tumours based on six extracted features from a dataset of mammograms. Figure created by Yue et al. (2018). . 15

2.8 Example of a typical CNN adapted for multi-class breast cancer detection. Figure adapted from S. Saha (`https://tinyurl.com/y9mmosuq`). 18
2.9 Difference between max pooling and average pooling using a 2x2 window and stride 2 (left) to downsample an image (right). Figure adapted from W. Ong (`https://tinyurl.com/y25cke6l`). 20
2.10 Example of transformations applied to a mammogram to generate new images. Original image retrieved from the mini-MIAS dataset (Suckling, 1994). 22
2.11 Example of standard neural network (left) and a neural network with dropout applied (right). Figure retrieved from Srivastava et al. (2014). 23

3.1 Types of structures and views captured by the CBIS-DDSM dataset (CC and MLO mammogram views are from the same patient). 27
3.2 The three different types of breast background found in the mini-MIAS dataset. 28

4.1 A high-level flowchart of the breast cancer detection deep learning pipeline to implement, separated into data pre-processing, model training, results visualisation and fine-tuning. 30
4.2 Class distribution for the mini-MIAS and the CBIS-DDSM datasets. 31
4.3 Example of the pixels values that make up a mammogram before and after normalisation. 33
4.4 CNN architecture used. VGG19 image retrieved from `https://tinyurl.com/rpp49oc`. 35
4.5 Visualisation of the sigmoid and softmax activation functions. . 35

5.1 Original dataset divided into training, validation and testing sets using a 60/20/20% split. 44
5.2 Example of affine transforms applied to mini-MIAS mammograms to generate new samples. Original image retrieved from the mini-MIAS dataset (Suckling, 1994). 45
5.3 A detailed flowchart of the breast cancer detection deep learning pipeline implemented, separated between data pre-processing, model training, results visualisation and fine-tuning. 48

6.1 Training (2445 samples) and prediction (641 samples) runtimes on the CBIS-DDSM dataset when using different CNN architectures as the base model pre-trained on ImageNet 51
6.2 Confusion matrix when double data augmentation is applied on the mini-MIAS test dataset with MobileNetV2 as the base model. 52

6.3 Training runtimes when using no data augmentation, augmentation to fix class balance and to double the training set size (744, 372 and 192 samples respectively) on the mini-MIAS dataset. 53
6.4 Normalised confusion matrix when no class weights are used with MobileNetV2 as the base model on the CBIS-DDSM dataset. 54
6.5 Normalised confusion matrix when balanced class weights are used with MobileNetV2 as the base model on the CBIS-DDSM dataset. 55
6.6 Evolution of the accuracy and loss during both training phases when testing 1024x1024 input size on VGG19. 56
6.7 Training (2445 samples) and prediction (641 samples) runtimes on the CBIS-DDSM dataset when using different input image sizes (224, 512 and 1024 pixels). 57
6.8 Training (2445 samples) and prediction (641 samples) runtimes on the CBIS-DDSM dataset when using different amounts of transfer learning through the binary mini-MIAS and ImageNet datasets with MobileNetV2 as a base model. 59
6.9 Training and prediction runtimes on the CBIS-DDSM dataset when using different mammogram types (all, masses and calcifications). 61
6.10 Bar chart summarising the relative accuracies achieved for each experiment compared to the baseline developed as group on the CBIS-DDSM dataset. 62

B.1 Chart highlighting the number of articles mentioning Keras and PyTorch. Figure downloaded from Keras website. 77
B.2 Chart depicting the number of PyPI downloads for Keras and PyTorch. Figure downloaded from Keras website. 77

D.1 Screenshot of the Jupyter Lab interface used to implement the project. 83

F.1 Screenshot of the project structure. 95

List of Tables

4.1 Conversion from string format (categorical) to one-hot encoding. 33
4.2 Conversion from string format (categorical) to binary encoding. 34

6.1 Results achieved on the CBIS-DDSm test set when using different CNN architectures as the base model pre-trained on ImageNet weights. 50
6.2 Results achieved on the mini-MIAS test set when using different amounts of data augmentation (none, balanced and double). . . 52
6.3 Results achieved on the CBIS-DDSM test set when using different class weights (none, balanced and +50% minority class) with VGG19 and MobileNet architectures as base model. . . . 54
6.4 Results achieved on the CBIS-DDSM test set when using different input image sizes (224, 512 and 1024 pixels). 56
6.5 Results achieved on the test set when using different amounts of transfer learning on the CBIS-DDSM dataset with the MobileNetV2 base model. 58
6.6 Results achieved with on the test set when using different types of mammograms (VGG19, ResNet50, InceptionV3, DenseNet121 and MobileNetV2) on the CBIS-DDSM dataset. 60

7.1 DDSM dataset patient population statistics (female). Data collected by Massachusetts General Hospital (MGH) and Wake Forest University School of Medicine (WFUSM) (Heath et al., 2001). 64

B.1 Table comparing the pros and cons of different programming languages when implementing a deep learning system. 76

List of Acronyms

Adam Adaptive moment estimation

ANN Artificial Neural Network

AR Association Rule learning

AUC Area Under the Curve

BCD Breast Cancer Detection

BP Backpropagation

CAD Computer-Aided Detection/Diagnosis

CBIS-DDSM Curated Breast Imaging Subset of DDSM (dataset)

CC Bilateral craniocaudal (mammogram)

CLI Command Line Interface

CNN Convolutional Neural Network

CPU Central Processing Unit

DT Decision Tree

GPU Graphical Processing Unit

kNN k-Nearest Neighbours

mini-MIAS mini Mammography Image Analysis Society (dataset)

MLO Mediolateral oblique (mammogram)

MLP Multi-Layer Perceptron

MRI Magnetic Resonance Imaging

NB Naive Bayes

OOM Out Of Memory (error)

PNN Probabilistic Neural Networks

RAM Random-Access Memory

ReLU Rectified Linear Unit

RMSProp Root Mean Square Prop

ROC Receiver Operating Characteristic

ROI Region of Interest

SGD Stochastic Gradient Descent

SVM Support Vector Machine

WBCD Wisconsin Breast Cancer Wisconsin (dataset)

Acknowledgements

I have received a great deal of support and guidance throughout my Master's degree at the University of St Andrews and would like to take the opportunity to thank those who helped me and motivated to always improve throughout this year.

First of all, I would like to thank my project supervisor, Dr David Harris-Birtill, whose expertise in the domain of machine learning has been invaluable to this project. I would like to extend my thanks to my project's co-supervisor, Lewis McMillan, for his practical knowledge of implementing deep learning systems, and to my team members, Ashay Patel and Shuen-Jen Shen, for their intuitive insights and help when developing the code in the middle of a pandemic. I also wish to thank the individuals who took the time, effort and patience to proof-read this dissertation and offer precious advice.

Finally, I would like to thank all my family and friends, especially my mother, father, sister and grandmother, for their love, encouragement and constant support to pursue my dreams.

Chapter 1

Introduction

1.1 Motivation

Breast cancer is one of the most common forms of cancer amongst women in the UK, with statistics indicating that 1 in 7 females will be diagnosed with breast cancer in their lifetime. Indeed, 55,200 new breast cancer cases are reported every year in the UK, of which a disheartening average of 11,400 lead to death (Cancer Research UK, 2020). With an average of 20% mortality rate, breast cancer is ranked as one of the deadliest diseases.

Early detection of breast cancer through screening tests such as mammograms is an efficient way to maximise patients' survival rate by treating the disease prematurely. However, no matter the expertise of radiologists examining mammograms, external factors such as fatigue, distractions and human error need to be minimised (Polat and Güneş, 2007), as the rate of missed breast cancers during initial mammogram screenings are as high as 30% (Elter and Horsch, 2009). To convey the complexity of mammogram interpretation, Figure 1.1 illustrates three different mammograms containing either normal or abnormal (benign and malignant) cases, and how similar they all look to an untrained eye.

Error-prone mammogram interpretations by radiologists can lead to decisions that can ultimately harm the patients. If a mammogram is diagnosed as malignant, breast biopsies (extraction of cells in breast tissue) are usually prescribed. However, 40-60% of biopsies are diagnosed as benign, clearly betraying the necessity for correct mammography diagnosis to avoid needless operations, anxiety and pain for the patients (Hepsağ et al., 2017). On the one hand, breast cancers can be missed altogether, inducing an absence of treatments for sick patients, while on the other hand, an instance of breast cancer can be reported when in reality there is no cancerous tumour, leading to unnecessary treatment being carried out (Elter and Horsch, 2009).

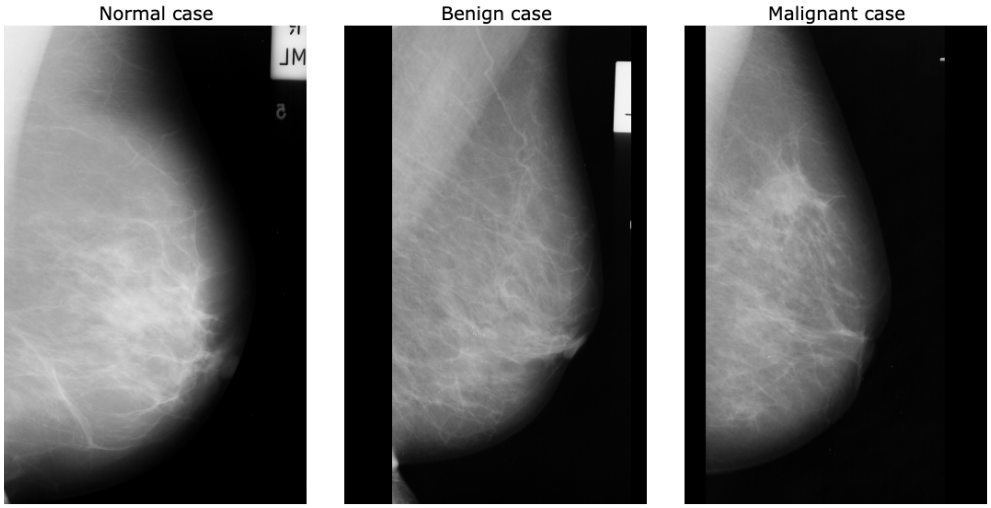


Figure 1.1: Example of three types of mammograms, including normal, benign and malignant cases. Figures extracted from the mini-MIAS dataset (Suckling, 1994). Created using draw.io.

To that end, using Computer-Assisted Detection (CAD) software can help minimise the number of wrong interpretations and increase the accuracy of mammography screening (Shen et al., 2017).

The motivation behind this project is to explore techniques for implementing a deep learning system that can accurately detect breast cancer in order to prevent late treatments due to false negatives as well as preventing unnecessary treatments in cases of false positives. Ultimately, the long-term target of this project is to combine it with other deep learning algorithms developed across other projects supervised by Dr David Harris-Birtill (past and present). This will allow a general artificial intelligence system capable of detecting multiple forms of cancer with higher accuracies than radiologist diagnoses.

1.2 Problem Description

CAD systems using deep learning techniques could, in theory, highly increase the accuracy of mammogram screenings for detecting early signs of breast cancers. However, these techniques require large amounts of data to learn the cancer's underlying patterns and adapt to new cases, and require powerful computing resources to accelerate the process of learning the data, making them very hard to optimise.

Parts of the work undertaken during this project will be conducted as a group comprised of two other members, Ashay Patel and Shuen-Jen Chen. Section 1.3 covers which tasks will be conducted personally/in a group in more detail. The reasoning behind these common tasks is for a functional pipeline to be reached earlier, eventually allowing each group member to further explore deep learning techniques individually more quickly due to the limited time frame of this project, using the common primary pipeline as a baseline.

1.3 Objectives

The main objective of this project consists of implementing a deep learning pipeline that will be able to learn how to detect cases of breast cancer in mammograms. This objective is broken down into two steps:

- **Group work**: a common deep learning pipeline will be initially implemented as part of a group with Ashay Patel and Shuen-Jen Chen over the course of a three-week period, including data cleaning and preprocessing, results output and a basic deep learning model. The distribution of tasks between the group can be found in Appendix E.
- **Individual work**: the pipeline above will then be individually extended and evaluated by using various deep learning techniques.

An extensive context survey has to first be conducted to cover the background of deep learning techniques applied to the field of cancer detection and to review existing results. This includes identifying results achieved using different methods (e.g. traditional machine learning techniques). This step is primordial as it will guide the research towards the most promising areas, as well as govern the choice of techniques to implement and explore in further chapters.

Finally, the final results achieved individually will be compared with the baseline pipeline created as a group, as well as the results found in papers that used the same datasets.

1.4 Report Structure

Introduction Presents an overview of the subject's background through the problem description and the motivation behind this project, followed by the objectives that the project aims to achieve.

Context Survey Explores the literature and background surrounding breast cancer detection techniques, starting from primitive cancer detection systems, followed by traditional machine learning methods, and ending with

the deep learning techniques that have been recently used.

Ethics & Datasets Considers the ethical issues taken into account for this project and describes the datasets used.

Design Explores high-level design considerations regarding the deep learning pipeline to implement and the software in general.

Implementation Comprehensively covers the steps followed when implementing the deep learning pipeline, explaining the practical solutions followed.

Evaluation Reviews the different results to assess the efficiency of the different techniques used to train the model and how it compares to other models, including the common pipeline, the baseline and relevant results identified in the context survey.

Conclusions Summarises the project's accomplished objectives, its limitations, plans for future work, and a final reflection on the project as a whole.

Chapter 2

Context Survey

2.1 Breast Cancer Detection

2.1.1 Medical imagery screening tests & biopsies

Test screenings have been used to detect early signs of breast cancer before the appearance of any symptoms (e.g. lumps that can be felt to the touch of a hand). The main methods used for breast cancer screenings are *mammograms*, which are low-dosage x-rays around the breast area usually used as initial/regular screening tests. These scans reveal backgrounds in black and dense areas in white, which may correspond to calcifications or masses (e.g. lumps or cysts). If suspicious areas are detected, mammograms are followed by *breast ultrasounds* for analysing these masses, and by *breasts MRIs* (Magnetic Resonance Imaging) for detailed imagery of the breast, usually used when a malignant tumour has been detected to get more information about it, such as its size and location, or to find additional ones (American Cancer Society, 2019). If any of the screenings mentioned earlier raise suspicion or reveal a potential presence of breast cancer, then biopsies can be conducted to confirm the screening tests' results. Biopsies consist of extracting cells or a small part of the breast's tissue and sending them to a lab to be analysed by pathologists to get definite results (Martin, Laura J., 2019).

Due to the invasive nature of biopsies, it is ideal for patients to use medical imagery tools to detect early signs of breast cancer that can be treated efficiently rather than immediately conducting a biopsy. Mammograms are the primary imagery method used for early breast cancer detection (BCD) (Ramos-Pollán et al., 2012). However, BCD using mammograms, and any form of cancer detection using medical imagery, relies on the conventional diagnoses of expert radiologists (Osareh and Shadgar, 2010). These diagnoses rest on the correct interpretation of the mammograms, which may be subject to errors due to the difficulty of correctly interpreting them (Elter and Horsch,

2009). Indeed, mammograms are 2D images of 3D breasts that correspond to the superposition of breast tissue, which increases the difficulty for a radiologist to correctly analyse patterns as masses often naturally form due to this superposition (Elter and Horsch, 2009).

2.1.2 Early Breast Cancer Detection Systems

To assist radiologists in their interpretations of mammograms, CAD software has been employed since the 1970s. However, pre-1990s CAD systems were very primitive and did not offer much more knowledge than the expert radiologists' knowledge. These unsophisticated "expert" systems consisted of manually processing and modelling pixels to construct rule-based systems that mainly used *if-else-then* statements (Litjens et al., 2017), highlighting their inadequacy to learn how to recognise patterns that can be used to detect the vast possible forms that breast cancer can take.

2.1.3 Towards Supervised Machine Learning-based Systems

Towards the late 1990s, supervised machine learning techniques started replacing these expert systems, allowing hidden patterns in the mammograms' data that could not be perceived by radiologists to now be recognised by these new algorithms (Litjens et al., 2017). Machine learning-based approaches were selected over statistical approaches to replace expert systems as they were proven to be more suitable for classification tasks than traditional statistics-based approaches such as regression (Paliwal and Kumar, 2009), especially when dealing with large, complex and high-dimensional datasets like mammogram datasets (Yue et al., 2018). This marked the shift from CAD systems that were fully designed by humans to systems that were trained on datasets of medical imagery (Litjens et al., 2017).

However, these machine learning models could not accurately operate on purely raw data such as the full-sized mammogram images. Indeed, all of the machine learning models tested against the task of BCD required relevant pieces of information to first be extracted from the image to solve the given task, using models such as k-Nearest Neighbour [kNN], Decision Trees [DT], Naive Bayes [NB] (Asri et al., 2016), Support Vector Machines [SVM] (Ramos-Pollán et al., 2012) and Artificial Neural Networks [ANN] (Yue et al., 2018). These crucial bits of information pulled from the mammograms data correspond to features, and need to be extracted by humans before being fed to the aforementioned models for training. These features range from visual information, such as colours, edges, corners, shapes and textures (Li and Allinson, 2008), to extracted information, such as the cell size, clump thickness, bare nuclei, etc. (Yue et al., 2018).

Logically, the next step in the evolution of BCD systems is for the model to learn these features on its own directly from the data, rather than being fed hand-crafted features (Yala et al., 2019). Deep learning models, which corresponds to neural networks with hundreds of hidden layers, are based on this concept. However, these models have not been successfully implemented until recent years as they require powerful computers, usually equipped with Graphical Processing Units (GPU) to be efficiently trained. This means that until recent years, machine learning models have led the field of BCD, with some manual mammogram interpretations still being carried out by radiologists (Litjens et al., 2017), as depicted in Figure 2.1.

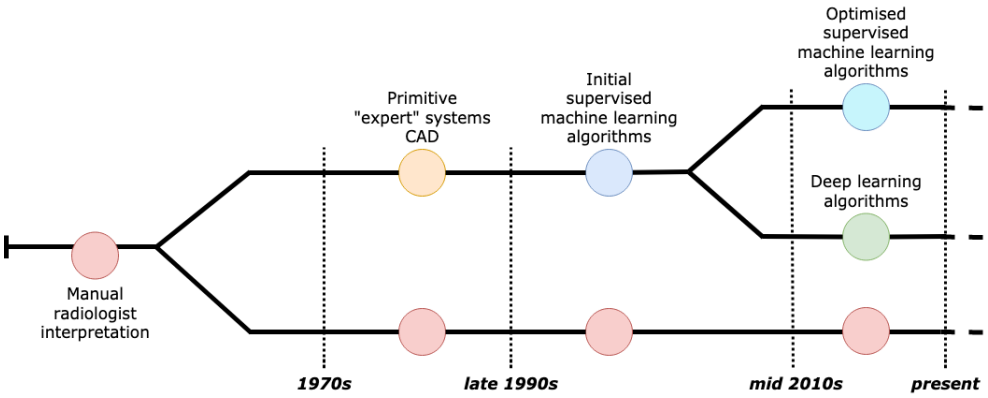


Figure 2.1: Timeline of the evolution of breast cancer detection (BCD) systems synthesising the information described in Sections 2.1.1 and 2.1.2.

2.2 Machine Learning Tasks & Algorithms

2.2.1 Machine Learning Applications to Breast Cancer Detection

Types of machine learning algorithms

Machine learning algorithms fall in different categories based on whether human supervision is required or not. The two main types of machine learning algorithms correspond to supervised and unsupervised learning. On the one hand, in supervised learning, the dataset is labelled, meaning every sample in the dataset includes a solution (Caruana and Niculescu-Mizil, 2006). This label y is used to make a prediction $\hat{y}$ by fitting the input features $\mathbf{x}$ from a training dataset. The goal of a supervised learning algorithm is to determine the optimal parameters θ for the selected algorithm in order to minimise a loss function defined as $L(y, \hat{y})$, which corresponds to the error between $\hat{y}$ and y (Litjens et al., 2017). A large variety of loss functions can be used such as the general Mean Squared Error (MSE) and Mean Absolute Error (MAE)

loss functions, or more specific loss functions such as the Hinge Loss for SVMs (Géron, 2019). The main applications of supervised learning are classification and regression, with the former being the most relevant to BCD.

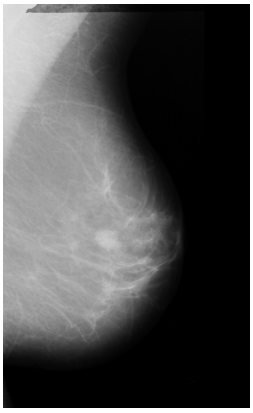
On the other hand, in unsupervised learning, the data is unlabelled, meaning only the input features $\mathbf{x}$ are available while the labels y are not (Litjens et al., 2017). This means the algorithm cannot optimise its hyperparameters, which correspond its configurations used to define it before training (Bergstra et al., 2013), by minimising a loss function. Instead, the algorithm needs to automatically create clusters in the dataset in order to separate them into different groups. The main applications of unsupervised learning are clustering, anomaly detection, data visualisation and dimensionality reduction (Géron, 2019), rendering them irrelevant to breast cancer detection. Two other categories of machine learning algorithms exist, corresponding to semi-supervised learning and reinforcement learning, but are also irrelevant to the task of detecting breast cancer.

Among the two types of machine learning algorithms, the most pertinent one for the task of BCD is supervised learning as datasets of mammograms need to contain properly labelled data for each sample, indicating the status of the mammogram, i.e. no tumour, benign tumour, malignant tumour (Shen et al., 2017).

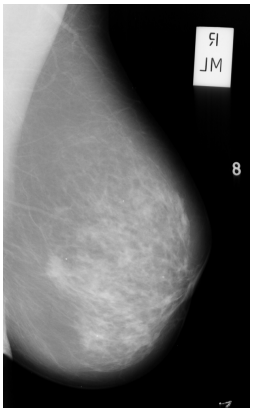
Types of machine learning tasks

Two types of machine learning tasks are relevant to medical imagery analysis, including mammogram analysis for BCD: *detection* (classification) and *segmentation* (Litjens et al., 2017).

Detection corresponds to the classification of a medical image or exam, which is an interpretation that used to be entirely carried out by a radiologist before the appearance of CAD systems. The classification can be either binary or multi-class, depending on the data used (see Section 4.1). With datasets like the "Curated Breast Imaging Subset of DDSM" (Lee et al., 2017), the classification is binary as mammograms can only be classified as either "benign" or "malignant". However, when using datasets like the "Digital Database for Screening Mammography" (Heath et al., 2001), the classification becomes more interesting from a clinical point of view as mammograms can either be classified as "normal", "benign" or "malignant" (Litjens et al., 2017). An example of classification between benign and malignant mammograms can be found in Figure 2.2, reinforcing the complexity that comes with interpreting mammograms for radiologists and the need for accurate and reliable CAD systems to improve the prediction accuracies.



(a) A benign mammogram.



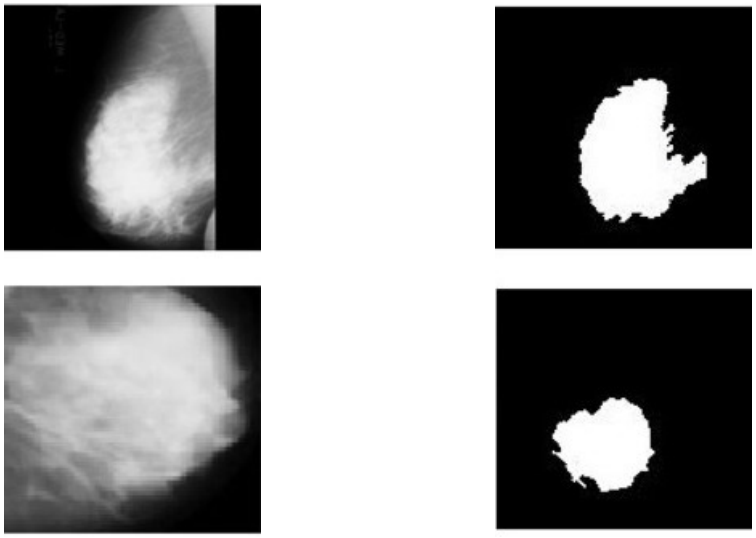
(b) A malignant mammogram.

Figure 2.2: Example of a breast mammogram classification, showing benign (left) and malignant (right) mammograms. Images retrieved from the mini-MIAS dataset (Suckling, 1994).

Segmentation corresponds to the classification of each pixel in the image based on the class the pixels belongs to, without distinguishing pixels from the same class. All objects in the image belonging to the same class will be classified in the same grouping of pixels (Géron, 2019). In breast cancer detection, segmentation can be used to highlight masses such as calcifications, cysts or fibroadenomas (*What Mammograms Show: Calcifications, Cysts, Fibroadenomas*, 2018) in mammograms by separating the masses from the background. An example is shown in Figure 2.3, where mammograms are partitioned into non-overlapping segments, revealing potential masses in the image (Punitha et al., 2018).

Other machine learning tasks that have been used in medical imagery analysis consist of content-based image retrieval (retrieving similar images from a database) or image enhancement (erasing obstructing elements from an image and increasing quality) (Litjens et al., 2017), but will not be further explored as they are not directly relevant to the task of breast cancer detection.

The task of classification (detection) can be used to interpret whether a breast is affected by cancer through the analysis of databases of mammograms, while the task of segmentation can be used to localise a tumour within a breast by finding regions that may correspond to masses, pinpointing potential danger areas that may lead to cancerous cells.



(a) Original mammogram images. (b) Segmented mammogram images.

Figure 2.3: Example of a breast mammogram segmentation, showing the original mammogram (left) and the segmented image (right), depicting large masses. Images retrieved from Punithaet al. (2018).

2.2.2 Comparison of BCD Supervised Learning Algorithms

Since the late 1990s, a rich array of supervised machine learning algorithms have been applied and tested against the task of BCD, contributing to improving accuracies for detecting breast cancer (Yue et al., 2018). The main types of algorithms used in BCD, which consist of k-Nearest Neighbour (kNN), Naive Bayes (NB), Support Vector Machines (SVM), Decision Trees (DT) and Artificial Neural Networks (ANN), are briefly explained in the ensuing sections from most simple to most complex. Their performances in BCD are then compared to draw a picture of the advantages and disadvantages that each method brings.

k-Nearest Neighbours

k-Nearest Neighbours (kNN) is often used as an initial benchmark when studying a dataset with no prior knowledge (Peterson, 2009). It is a non-parametric and lazy model, as it does not learn the data's pattern but rather classifies a test sample by looking at its k nearest neighbours (Yue et al., 2018). The sample data point's nearest neighbours are determined by using distance metrics such as the Euclidian distance, which is the most widely used metric (Peterson, 2009). Equation 2.1 (adapted from Russell and Norvig (2002)) calculates the distance between two data points s and p in n-dimensional space. An example of how a kNN classifier would be used to distinguish between a benign and a

malignant tumour using $k = 3$ is depicted in Figure 2.4.

$$d(s, p) = \sqrt{\sum_{i=1}^{n} (s_i - p_i)^2} \tag{2.1}$$

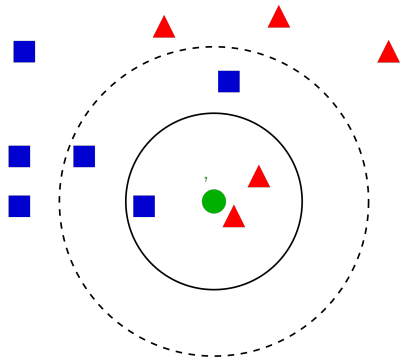


Figure 2.4: Example of a kNN classifier distinguishing between benign (blue square) and malignant (red triangle) tumours for a test data sample (green circle) using $k = 3$. The test sample is classified as malignant as there are two red triangles and one blue square amongst the three neighbours. Figure retrieved from T. Srivastava (`https://tinyurl.com/y3jqco49`).

kNN has been tested on datasets of mammograms such as the "Wisconsin Breast Cancer Wisconsin" (WBCD) dataset, which contains ten extracted features such as clump thickness, cell size/shape, etc. (Wolberg et al., 1995). kNNs calibrated with $k = 1$ and using the Euclidian distance achieved 98.25% accuracy (Sarkar and Leong, 2000) and 98.70% accuracy (Medjahed et al., 2013) on the WBCD dataset when compared with other values of k ranging from 1 to 15 and using other distance metrics such Cityblock distance, cosine distance and correlation. Despite Sarkar et al. and Medjahed et al. reporting that a value of $k = 1$ seemed to yield the most accuracy, these 1-NN classifiers do not actually learn the data's patterns and often underperform compared to other classifiers mentioned below, especially compared to SVMs and ANNs which can surpass accuracies of 99% (Yue et al., 2018; Asri et al., 2016; Montazeri et al., 2016). Nevertheless, the results achieved by kNN remain fruitful when used as a primary benchmark before testing more advanced models, such as the ones described below.

Naive Bayes

Naive Bayes uses Bayes' theorem and the assumption that all input features are independent from one another, which can be described as the input features

$\mathbf{x} = (\mathbf{x}_1, ..., \mathbf{x}_n)$ being independent given a class label C in Equation 2.2 (Rish et al., 2001).

$$P(\mathbf{x}|C) = \prod_{i=1}^{n} P(\mathbf{x}_i|C) \tag{2.2}$$

This assumption leads to a naive model that despite not learning the data's underlying pattern (in a similar fashion to kNN), still offers competitive results in practice (Russell and Norvig, 2002), notably in the field of medical imagery analysis (Rish et al., 2001). Naive Bayes achieves an accuracy of 93% on the WBCD dataset (Kharya et al., 2014), which is comparable to the aforementioned 1-NN classifiers as it does not learn the data's patterns and use extracted features rather than the original mammograms. NB still remains useful for assessing benchmark classification to compare with more advanced models.

Decision Trees

Unlike kNN and NB, the decision tree algorithm is a simple yet powerful one for fitting data. It works like a flowchart mapping samples' input feature vectors attributes and values, creating a tree made up of different types of nodes. Each non-leaf nodes tests one of the feature vectors attributes, branching out to a deeper node based on the attribute's value. Once a leaf node is reached after multiple tests, a classification decision is made (Quinlan, 2014). An example of a decision tree applied to breast cancer detection can be found in Figure 2.5, illustrating how attributes and their values are used to classify a tumour as benign or malignant.

The most popular implementations of decision trees use entropy-based impurity metrics to generate the tree, such as the ID3, C4.5 and C5.0 decision tree algorithms (Yue et al., 2018). A node reaches an entropy value of 0 when it is "pure", i.e. it contains only instances of one class (either only benign or only malignant data samples).

The extra complexity offered by decision trees such as the C4.5 classifier does do not offer improved results compared to kNN and NB classifiers, with 94.56% achieved using J48 (Java implementation of C4.5) on the WBCD dataset (Sumbaly et al., 2014), which is still falling short of the performance achieved by SVMs and ANNs that exceed 99% (Yue et al., 2018). Nevertheless, inserting decision trees into hybrid systems by combining them with other machine learning algorithms such as SVMs and NBs increases the accuracy to 97.13% on the WBCD dataset (Kumar et al., 2017), which is closer to those achieved by SVMs and ANNs (Yue et al., 2018). Indeed, these hybrid systems use the advantages of each method, cancelling out the negatives of

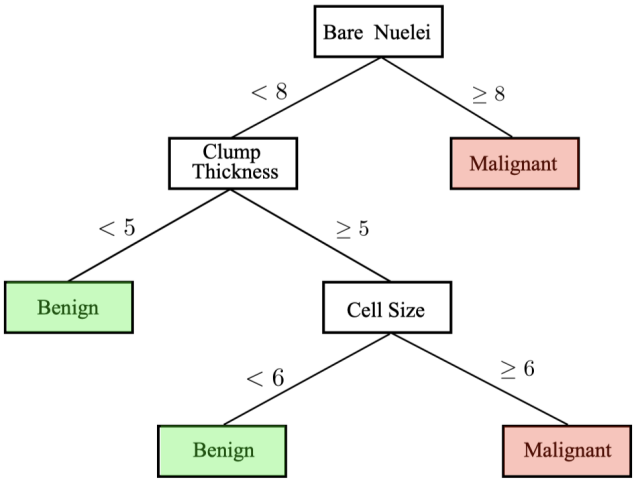


Figure 2.5: Example of a decision tree classifier distinguishing between benign and malignant tumours based on three extracted features from a dataset of mammograms: the size of the bare nuclei, the thickness of the clump and the uniformity of the cell size. Figure created by Yue et al. (2018).

each, but come at the cost of being complicated to engineer and still requiring hand-crafted features to be fed to them as input, which is confirmed by their decision of testing these algorithms on the WBCD dataset rather than datasets containing raw images like DDSM.

Support Vector Machines

An SVM consists of a *maximal margin classifier*, which aims to find the hyperplane that separates two classes the most, and the *kernel trick*, used to separate non-linear data. In Figure 2.6, a visual example of a maximum margin hyperplane for linearly separating benign and malignant tumours is shown.

In the case of non-linear data, the training data is mapped to a higher dimension where they can be linearly separated. This is achieved by using the kernel trick, which maps the input feature vector to a higher dimension by using the dot product, but does not carry out the transformation, which would exponentially increase the size of the feature space and consequently the training time (Géron, 2019). This kernel trick is what allowed SVMs to become one of the most widely used machine learning models in many fields, including medical imagery analysis, up until today (Yue et al., 2018).

Many kernels can be chosen for SVM classification, ranging from Polynomial kernel for image processing to Radial Basis Function (RBF) to Gaussian kernels for general tasks when there is no prior knowledge of the data, heavily

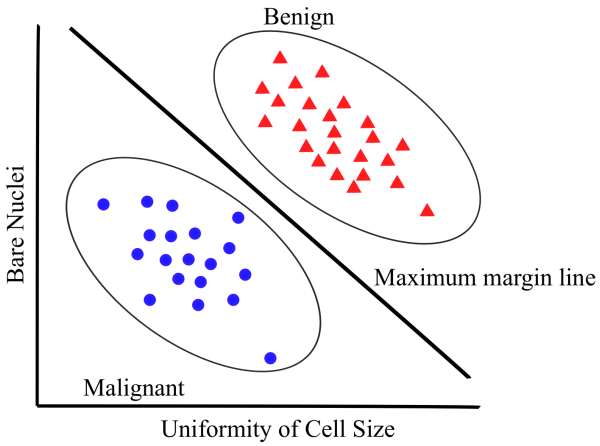


Figure 2.6: Example of a SVM classifier's maximum margin hyperplane found to separate benign and malignant tumours based on two extracted features from a dataset of mammograms: the size of the bare nuclei and the uniformity of the cell size. Figure created by Yue et al. (2018).

influencing its performance (Amari and Wu, 1999). For the task of breast cancer detection, RBF outperformed Polynomial kernels on two small datasets[1] containing extracted features, with the RBF kernel managing 98.80% and 96.33% accuracies on both datasets while Polynomial reached 97.09% and 95% accuracies (Osareh and Shadgar, 2010), depicting why RBF is often chosen over other kernels. Compared to the previously mentioned ML methods, SVMs seem to learn the extracted features more efficiently, reaching 97.13% accuracy on the WBCD, while kNN attained 95.27%, NB 95.99% and DT 95.13% (Asri et al., 2016).

When applied to datasets containing raw images such as CBIS-DDSM, SVMs coupled with feature extraction techniques such as grey-level co-occurrence matrices (GLCM) achieved 63.03% accuracy (Sarosa et al., 2018); while on smaller datasets like mini-MIAS, SVMs coupled with texture-based features extracted from pre-processed images resulted in 92% accuracy (Vishrutha and Ravishankar, 2014). An accuracy of 97.2% was achieved using the simplest form of SVMs on the WBCD dataset (Bennett and Blue, 1998), while 99.51% accuracy was reached with SVMs on the same dataset when selecting the five best features (Akay, 2009), clearly characterising the importance of feature selection and image processing techniques for ML algorithms such as SVMs.

[1] "*Fine needle aspirate of breast lesions*" (FNAB) dataset and "*gene microarrays*" dataset (Osareh and Shadgar, 2010).

Artificial Neural Networks

ANNs form the basis of contemporary deep learning and are at the heart of the techniques mentioned in Section 2.3. Originally inspired by the neuron connections found in the human brain (McCulloch and Pitts, 1943), ANNs correspond to a collection of neurons (units) that "fire" an output if the linear sum of its weighted inputs tops a threshold. These neurons are placed in hierarchical layers (input, hidden and output layers) which are connected via weighted links, leading to a *fully connected* neural network when all the neurons from one layer communicate with all the neurons in the following layer. Input layer neurons accept the feature vectors $\mathbf{x}$, hidden layer neurons process the data and output layer neurons represent an outcome for the classification (Russell and Norvig, 2002). Each neuron's output is determined by its activation function, which is usually a non-linear function like step, sigmoid, Tanh or ReLU (chaining linear functions results in a linear function) (Litjens et al., 2017). Figure 2.7 depicts an example fully-connected ANN used to classify benign and malignant tumours using six input neurons, eight hidden neurons and one output neuron.

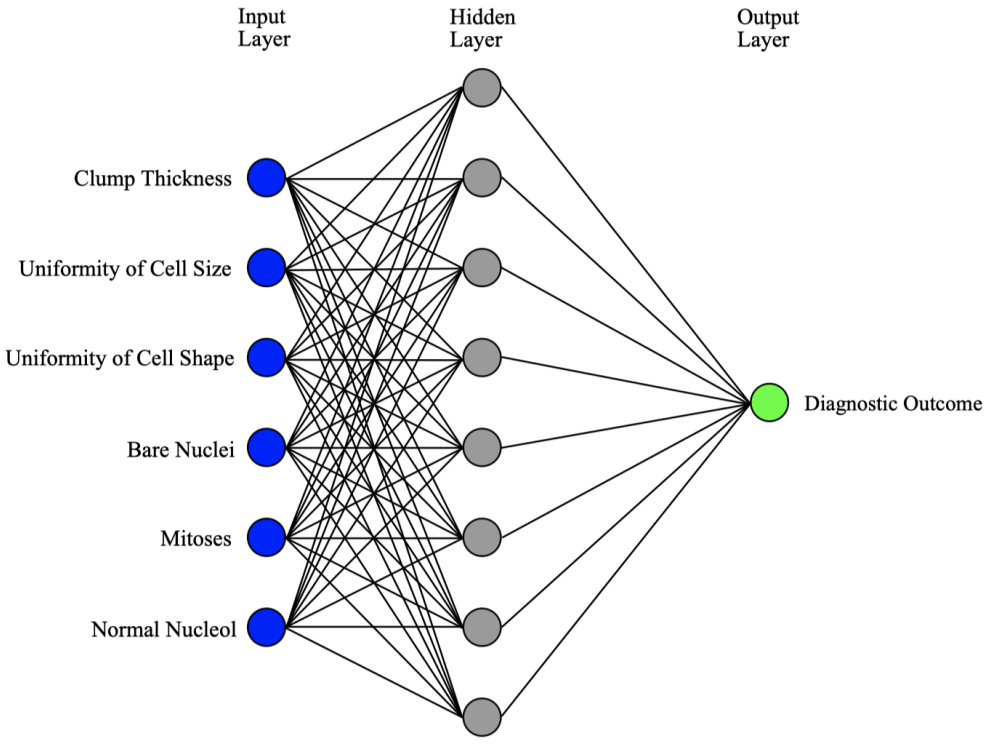


Figure 2.7: Example of an ANN classifier distinguishing between benign and malignant tumours based on six extracted features from a dataset of mammograms. Figure created by Yue et al. (2018).

Like the previous algorithms, neural networks learn by minimising the loss L between the prediction $\hat{y}$ and the labels y. This is done via the backpropagation algorithm (BP), which propagates the error from the output layer back to the hidden layers for each sample, allowing the links' weights to be adjusted accordingly to minimise that error (Russell and Norvig, 2002). A cycle of all the training samples is referred to as an epoch. The most common way to do this is by using a loss function with Stochastic Gradient Descent (SGD) (Litjens et al., 2017). This learning ability found in ANNs is what gives them their potential, but also their complexity due to the large number of hyperparameters used to fine-tune them and their capacity to overfit the data when over-engineered. Indeed, combinations of neural networks hyperparameters may involve the network's structure (number of layers and neurons in each layer), the learning rate and momentum for SGD, the regularisation techniques and parameters, the activation functions and the stopping conditions to name a few (*Scikit-Learn Documentation: Neural network models (supervised)*, 2019).

Shallow ANNs, also referred to as Multi-Layer Perceptrons (MLP), immediately showed promising results when first applied to breast cancer detection, reaching 95.2% accuracy with only a basic 3-layer architecture using BP (Wu et al., 1993), which already surpassed experts' predictions by 3-5% accuracy (Yue et al., 2018). However, due to computational constraints, deeper ANNs containing more hidden layers could not be efficiently utilised until recent years (Litjens et al., 2017). Instead, innovative variants to basic ANNs have been used to reduce training times, deal with multi-class classification with more ease, and yield predictions with higher accuracies. Some of these ANN variants include networks such as Probabilistic Neural Networks (PNN) that replace the standard sigmoid activation function with an exponential function to find the training sample that is closest to the testing sample, achieving 97.23% and 93.39% accuracies on the two small datasets of extracted features (mentioned in Section 2.2.2), which are larger than those achieved by kNN and Polynomial SVMs (Osareh and Shadgar, 2010). Another approach consisted of combining ANNs in hybrid systems (similar to those mentioned in Section 2.2.2). When combined to Association Rule learning (AR) for reducing the number of features in the WBCD dataset from 9 to 5, ANNs and AR reached 95.6% accuracy, which is extremely similar to the basic ANN on its own, but in almost half the epochs, converging in 33 epochs rather than 61 epochs, thus considerably reducing training time (Karabatak and Ince, 2009). Following that mindset of reducing the complexity of the WBCD dataset's features, Genetically Optimised ANNs (GOANN) use genetic programming (an ML technique that evolves towards a solution based on Darwin's Theory of Evolution) to determine the best features to use from the WBCD dataset and to optimise the network's weights and structure, producing an impressive

accuracy of 99.26% (Bhardwaj and Tiwari, 2015).

Supervised machine learning algorithms comparison

The five previously explored supervised machine learning algorithms all have one commonality: they heavily rely on the quality of the features extracted from the mammogram images as input to gain performance, and they cannot make use of the raw mammogram image in 2D space as input. This is confirmed by the datasets used for the task of BCD, as only datasets containing extracted features such as WBCD were used.

On their own, each algorithm showed limitations that prevent it from performing well. However, combined to form hybrid systems (e.g. DT + SVM + NB or ANN + AR), their performance increased in the form of higher accuracies or shorter training times, but so did their complexity to tune. It is worth noting that the slight differences when using identical algorithms on the same datasets can be due to the diverse training strategies involving unique training/testing/validation splits, image pre-processing steps and number of folds in cross-validation (Yue et al., 2018).

The next step for these supervised learning algorithms is to move away from feature extraction and redirect the effort towards new models that can automatically extract features from images rather than optimising and fine-tuning the hyperparameters and training strategies of existing algorithms. The most efficient way nowadays to achieve this is through Convolutional Neural Networks (CNN), which ingest the raw mammogram images in 2D space rather than extracted features or flattened images (transformed from 2D to 1D) where all spatial information is lost.

2.3 CNNs & Deep Learning techniques

2.3.1 Convolution Neural Networks

Motivation for CNNs over traditional neural networks

CNNs are a type of neural network inspired by the human visual cortex, where neurons have local receptive fields that only react to visual stimuli originating from a region of the visual field. The combination of all receptive fields covered by overlapping neurons forms the whole visual field (Li and Allinson, 2008). This architecture makes them very efficient at performing complex visual tasks, marked by the first milestone for CNNs with the LeNet-5 architecture trained to recognise handwritten bank cheque digits (LeCun et al., 1998).

CNNs differ from traditional "shallow" ANNs as they are not fully connected. Indeed, CNNs are partially connected, with neurons in one layer only connected to a few neurons from the previous layer, meaning that they can work with large images (Géron, 2019). CNNs nowadays can work with images thousands of pixel large, including the ones found in mammogram datasets described in Chapter 3, processing them much faster than traditional machine learning methods. In a fully-connected neural network with only 100 neurons in the first layer, a 1,000 x 1,000-pixel image would already have 100,000,000 connections[2] in that first layer alone.

CNN structure

The structure of CNNs builds on top of the concepts of traditional ANNs by piling stacks of convolutional layers and pooling layers that are followed by a shallow ANN for classification (see Figure 2.8). The goal of the convolutional and pooling layers is to reduce the input images into a form that is simple enough to be processed by the fully connected layers, retaining only useful information from the original image (Shen et al., 2017).

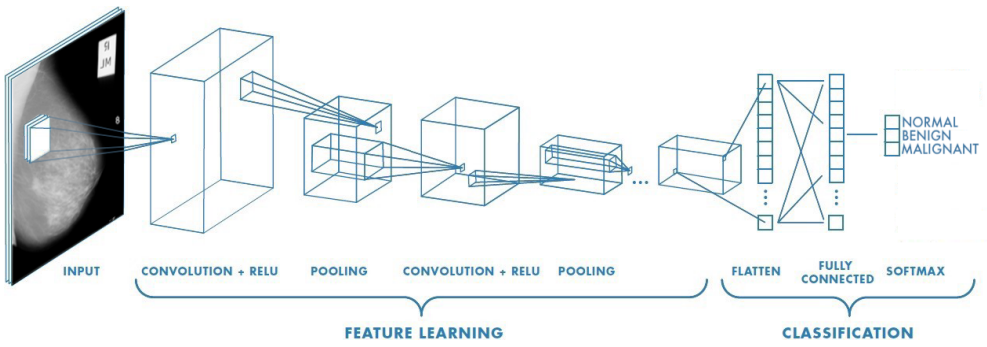


Figure 2.8: Example of a typical CNN adapted for multi-class breast cancer detection. Figure adapted from S. Saha (`https://tinyurl.com/y9mmosuq`).

Convolution Layers Neurons in the first convolutional layers are only connected to pixels in their receptive fields and not connected to every pixel in the image. Similarly, neurons in deeper convolutional layers are only connected to neurons in a small zone from the previous layer. This allows CNNs to first focus on low-level features, which are progressively assembled into high-level features as they get deeper. The more the receptive fields are spaced out (referred to as the stride), the smaller the next layer will be, thus considerably reducing the complexity of the CNN (Shen et al., 2017). Convolution is the mathematical operation that slides a moving a filter f over an image I to calculate a weighted sum (see Equation 2.3, Szeliski (2010)). This 2D operation

[2] $1000 \cdot 1,000 = 1,000,000$ px; $1,000,000$px $\cdot 100$ neurons $= 100,000,000$ connections.

is possible in CNNs as layers are represented in 2D space and do not need to be flattened into a 1D array like with traditional neural networks, thus preserving the spatial information of images (Szeliski, 2010).

$$\hat{I}(x,y) = (I * f)(x,y) = \sum_k \sum_l I(k,l) \cdot f(x-k, y-l) \tag{2.3}$$

The weights of the neurons in convolutional layers correspond to the filters, which are learned during the training phase by using optimisation techniques such as gradient descent. These filters allow a layer of neurons to highlight the areas of the image that activate the filter the most. As each layer has multiple filters, different features can be simultaneously detected in the layer's input. The result of each filter on a convolutional layer's input (the output of the previous layer) corresponds to a feature map. These multiple feature maps are all stacked together to form the convolutional layer's output.

Pooling Layers Pooling layers are similar to convolutional layers as neurons are only connected to neurons from a small region in the previous layer. The difference is that this layer is not trainable as its neurons have no weights. Indeed, it is only used to downsample the image as it traverses the network to diminish the load on the GPU. It does so by calculating an aggregate of its inputs based on a function, which can either be a maximum or an average function (see Figure 2.9). *Maximum pooling* returns the maximum value of the covered portion of the image, whereas *average pooling* returns the average of all values. Maximum pooling is the preferred option in CNNs as it retains the most robust features only and acts as a noise suppressant by discarding noisy activations (Krizhevsky et al., 2012).

Other benefits of pooling layers are the lower memory usage and the number of trainable parameters linked to smaller images, and especially the invariance it provides to the CNN as it will not break down when fed images that contain features of different sizes than the ones seen in the training dataset (Shen et al., 2017).

Fully connected layers & Activation functions Similarly to ANNs, activation functions are used to connect the convolutional and pooling layers. The most common activation function used in CNNs is the Rectified Linear Unit (ReLU).

At the end of the stack of convolutional and pooling layers, a fully connected MLP is placed. This dense neural network takes the flattened output of the stacked convolutional and pooling layers (which are transformed from 2D to 1D) and performs the classification tasks by using the features learned by the convolutional layers in a condensed format. Depending on the number of

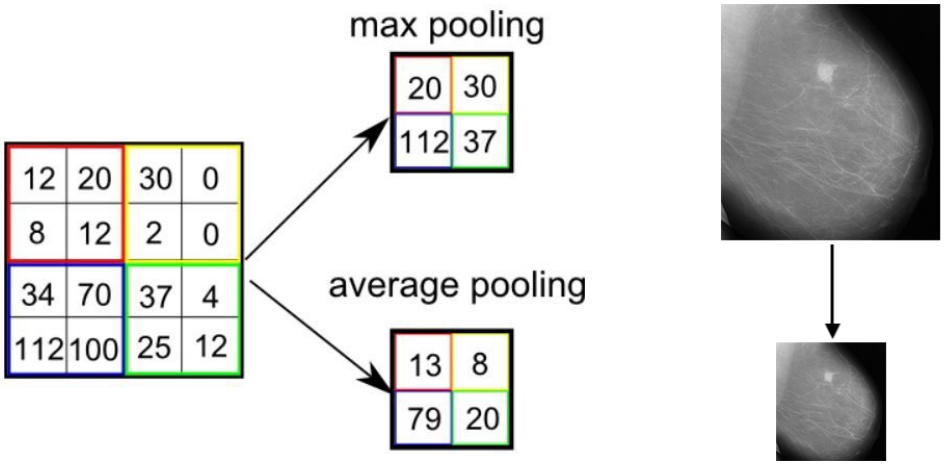


Figure 2.9: Difference between max pooling and average pooling using a 2x2 window and stride 2 (left) to downsample an image (right). Figure adapted from W. Ong (`https://tinyurl.com/y25cke6l`).

classes to predict, a softmax activation can be used for multi-class classification or a sigmoid for binary classification.

CNN Architectures

Deep CNN models such as AlexNet and VGG, which have won the *ImageNet Large Scale Visual Recognition Challenge*[3] in 2012 and 2014, remain very popular nowadays, especially in domains like medical imagery analysis where accuracies of 97.9% have been reached in BCD using VGG16 (Wang et al., 2016).

AlexNet is a CNN made up of five convolution layers that differs from traditional CNNs as not all convolution layers are separated by pooling layers (Krizhevsky et al., 2012). VGG architectures followed this concept by stacking multiple convolution layers with smaller filter sizes to pick up more complex features. Multiple variants of VGG exist based on the number of layers (VGG16 has sixteen layers and VGG19 has nineteen layers), but due to its depth, these takes a long time to train and run the risk of vanishing gradients (Simonyan and Zisserman, 2014). These are caused by the backpropagration algorithm coupled with gradient descent that lead to an exponentially smaller gradient while going back up the initial layers, which eventually prevents the network from learning as the weights and biases are no longer updated (Russell and Norvig, 2002). This problem does not occur in shallow ANNs.

[3]ImageNet challenge is a competition used to measure the performance of CNNs: `http://image-net.org/`.

More complex architectures were created subsequently to avoid creating deeper networks, such as ResNet, which uses residual modules, GoogLeNet, which uses inception modules, and MobileNet, which aims at maximising accuracy with little computing resources available; but are not covered in this context survey as efficiency gains is not an important factor in medical data, only the accuracy of the predictions matters (Litjens et al., 2017).

2.3.2 Deep Learning Applications in Breast Cancer Detection

Main challenges

Implementing deep learning models requires lots of data to achieve acceptable performance levels. However, labelled datasets are not always abundantly available as they require large amounts of time and computing resources to engineer large datasets with millions of images (Krizhevsky et al., 2012) like ImageNet, which contains over 15 million high definition images from 15,000 different classes (Deng et al., 2010). With databases of mammograms barely exceeding 10,000 images, one of the challenges of implementing a deep learning CAD system is to gain access to the computing resources needed to process the data and implement the model, as well as overcoming the problem of small amounts of data while avoiding overfitting it. Overfitting occurs when the model learns the data too well (e.g. detail and noise) and does not generalise well to new cases as it only recognises cases it has seen (Dietterich, 1995).

Transfer learning

A commonly used deep learning technique when only a little amount of data is available is transfer learning, which makes use of CNN models pre-trained on large general datasets. The knowledge gathered by high-performing CNNs in other general domains that have larger datasets can be transferred to a related domain such as medical imagery (Falconi et al., 2019).

These models are designed to classify millions of images across thousands of different classes and can easily be adapted to any classification task by replacing the dense output layer that makes the actual prediction with a layer containing one neuron per class to predict. Falconi et al. demonstrated how transfer learning from general domain datasets such as ImageNet could be transferred to the domain of mammograms using datasets such as CBIS-DDSM, reaching accuracies of 78.4% with the ResNet-50 model (Falconi et al., 2019).

Shen et al. showed how using common CNN architectures such as VGG or ResNet pre-trained on ImageNet resulted in accuracy increases. For instance, the accuracy improves by 2-27% based on the number of patches used (Shen

et al., 2017), while Diaz et al. demonstrated how two ResNet-50 models were tested with different weight initialisation: one on ImageNet weights and the other with random weights using the CBIS-DDSM dataset. The model using transfer learning achieved 84% accuracy compared to 75% accuracy (Diaz et al., 2018), clearly depicting the advantage of using CNNs with pre-trained weights.

Regularisation techniques

The drawback of the power showcased by deep neural networks such as CNNs is their tendency to overfit the data. To this end, new regularisation techniques were introduced to prevent overfitting.

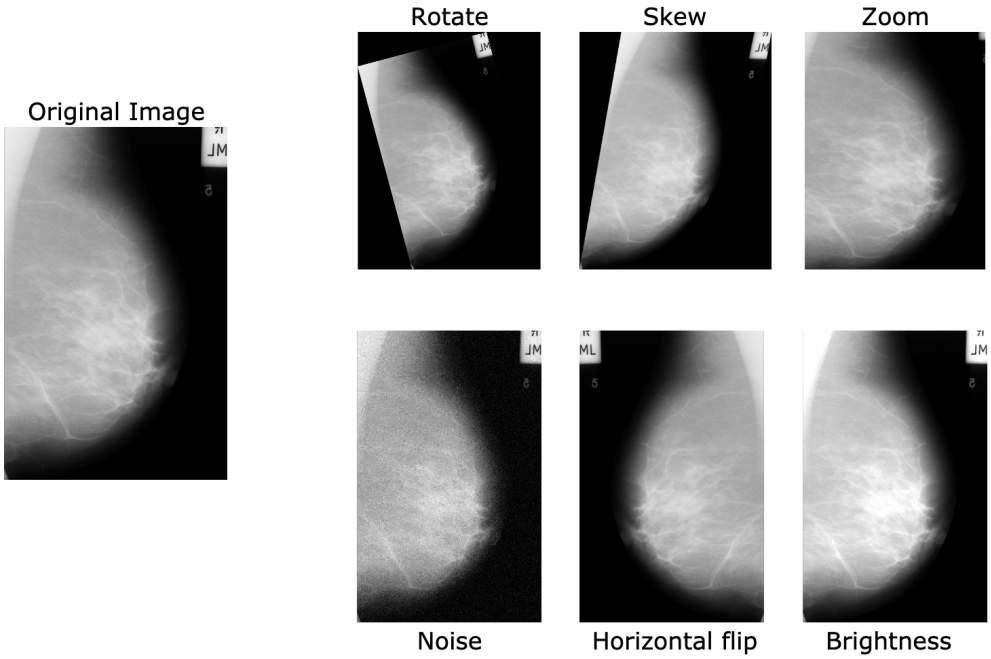


Figure 2.10: Example of transformations applied to a mammogram to generate new images. Original image retrieved from the mini-MIAS dataset (Suckling, 1994).

Data augmentation Another technique to counter small datasets is data augmentation, often used when attempting to learn a small dataset using complex deep learning models that have a large number of parameters, which may naturally lead to overfitting (Jadoon et al., 2017). The data is "augmented" by artificially creating similar realistic variants of the images found in the training set, considerably increasing the training set size. A varying amount of transformations can be applied to each image such as translation, rotation,

scaling, shear, horizontal/vertical flips, brightness and contrast increases (see Figure 2.10) (Hepsağ et al., 2017). Chen et al. show how applying data augmentation using affine transformations increases the accuracy from 83.6% to 88.14% using a ResNet-50 model (Chen et al., 2019).

Dropout Simply implementing dropout, the most popular regularisation technique for deep neural networks, in any CNN has been proven to boost the accuracy by 1 to 2% (Géron, 2019) at the cost of training time, even for state-of-the-art neural networks tested on large datasets like ImageNet (Srivastava et al., 2014). Despite training times being increased by 2 to 3 times, the gains in accuracy compensate for the extra time required to train the models implementing dropout.

Dropout works by randomly ignoring neurons in any layer (except the output layer) during the forward and backward passes of training, including its input and output connection weights, which helps prevents neurons from co-adapting too much with their neighbours and overfitting the data as they now have to be useful on their own. Essentially, new thinner networks are created at each training step (see Figure 2.11), and identical networks will never be sampled in the same training phases as there are 2^N possible networks, where N is the number of dropable neurons. The number of neurons dropped in a layer is controlled by the dropout rate hyperparameter p, dictating the probability of a neuron being dropped. During testing, the neurons are no longer dropped, and an averaged ensemble of all the thinner trained networks is used. This leads to models that generalise better thanks to neurons that are less sensitive to noise and small changes in the input (Srivastava et al., 2014).

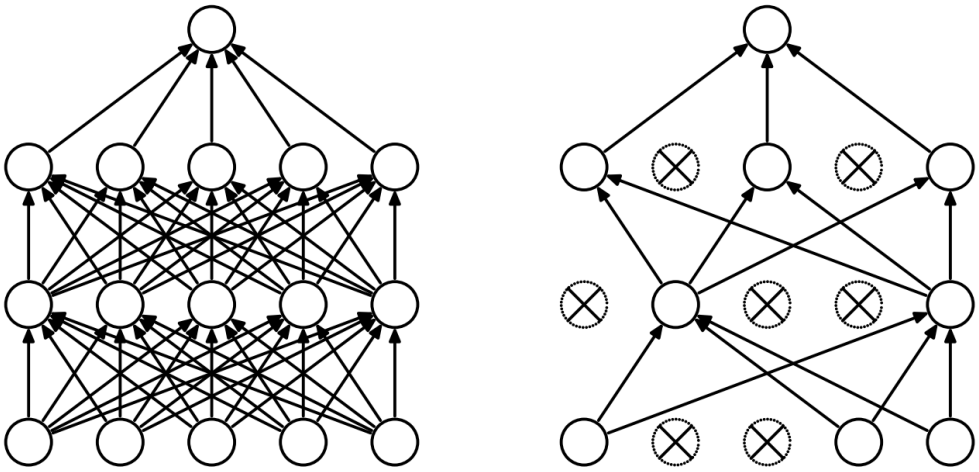


Figure 2.11: Example of standard neural network (left) and a neural network with dropout applied (right). Figure retrieved from Srivastava et al. (2014).

Technological advances

The main contributor to the rise of deep learning is linked to the large scale spread and availability of Graphical Processing Units (GPU) in recent years. GPUs are much more powerful than the general-purpose Central Processing Units (CPU), as they can process large amounts of data in parallel. Initially designed for computer graphics in video games, GPUs are extremely efficient for large-matrix operations, rendering them vital in the field of deep learning (Caulfield, 2009). GPU-computing libraries such as CUDA or OpenCL make it possible to use the processing power of GPUs, leading to computing times that are 10 to 30 times faster than CPUs (Litjens et al., 2017).

However, GPUs are complicated to use efficiently from a low-level perspective, which is why the collection of open-source software libraries available online has been essential for the rise of deep learning. These libraries allow high-level GPU-efficient implementations of the most important deep learning methods and operations, allowing focus to be placed on efficiently implementing deep learning pipelines rather than low-level GPU optimisations (Litjens et al., 2017). The most popular packages nowadays include Tensorflow (Abadi et al., 2015) coupled with Keras (Chollet et al., 2015) and PyTorch (Paszke et al., 2019).

Implementing CNNs entails another drawback linked to the memory requirements (RAM) and the use of GPUs. Indeed, input data, neuron weights and biases all need to be stored in memory as the data propagates through the neural network, especially during training as the data needs to be retained during backpropagation to calculate the error gradients (Géron, 2019). For GPU optimisation, this data is formatted as dense vectors for parallel processing optimisations, which can increase local RAM requirements to over 7.5GB for a CNN like ResNet-50 (Hanlon, 2016).

Without the rise of popularity in both GPUs and software libraries to take advantage of the additional processing power offered by GPUs, and the increase in other computing resources such as the amount of GPU and RAM, deep learning would not have been successfully applied to fields such as medical imagery analysis nowadays.

2.4 Summary

CAD systems have been developed since the 1970s to assist radiologists in their interpretations of mammograms, starting with primitive expert systems. These systems were replaced by supervised machine learning algorithms but required hand-crafted features to be extracted from the images. The perfor-

mance of the algorithms heavily relied on the quality of the features, which could not operate on raw mammograms. Therefore, deep learning algorithms have gained traction recently, notably CNNs, to automatically learn which features to extract from the images, thus preserving their 2D spatial property. However, CNNs require large amounts of data to avoid overfitting, which is not always available, especially in medical imagery.

Chapter 3

Ethics & Datasets

3.1 Ethical Considerations

The deep learning model aimed to be implemented during this project will require real-life data to learn the underlying patterns in order to be able to detect cases of breast cancer in mammograms and to evaluate its performance. Therefore, the main ethical concern when using sensitive medical data such as mammograms is whether it can be traced back to the original patient. As a result, fully anonymous, open-source and public datasets are used. A full Ethics Application to the University of St Andrew's Teaching and Research Ethics Committee was therefore submitted at the early stages of the project and later approved. The approval letter from the committee can be found in Appendix A.

3.2 Datasets Description

Three open-sourced, anonymised public datasets have been approved by the University of St Andrew's Teaching and Research Ethics Committee for use in this project; the DDSM, CBIS-DDSM and mini-MIAS datasets.

3.2.1 DDSM

The "Digital Database for Screening Mammography" (DDSM) dataset is a dataset initially released in 2007 and available online from the University of Florida. It holds 2,620 scanned film mammographies of normal, benign and malignant cases, all stored in Lossless JPEG format (LJPEG), which is obsolete nowadays (Heath et al., 2001).

3.2.2 CBIS-DDSM

The "Curated Breast Imaging Subset of DDSM" (CBIS-DDSM) dataset (Lee et al., 2017) is available online from The Cancer Imaging Archive (Clark et al., 2013). The dataset contains a total of 10,239 images in Digital Imaging and Communications in Medicine format (DICOM) gathered from 1,566 patients across 6,775 studies (Lee et al., 2017). This dataset is an updated and standardised subset of the older DDSM dataset (Heath et al., 2001), containing only abnormal cases with benign and malignant tumours (no normal cases).

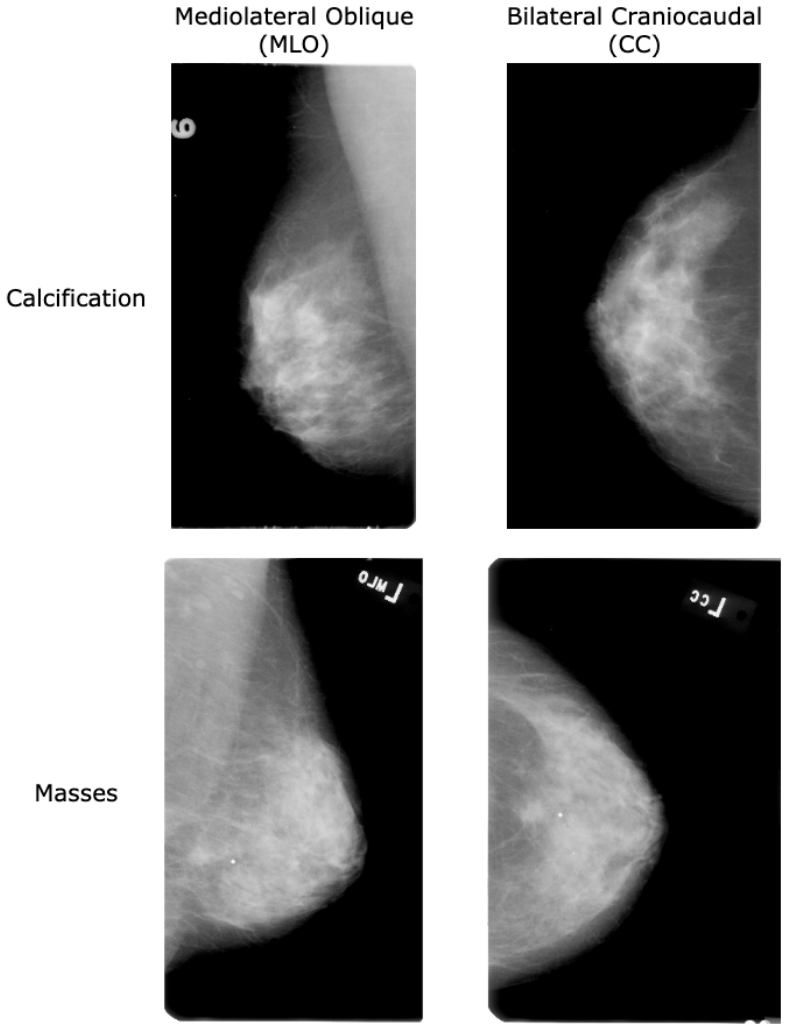


Figure 3.1: Types of structures and views captured by the CBIS-DDSM dataset (CC and MLO mammogram views are from the same patient).

The scans are a mix of the two most commonly used projections in routine mammogram X-ray scans: bilateral craniocaudal (CC) and mediolateral

oblique (MLO) (Elter and Horsch, 2009). The dataset can be further be separated into two different types of structures that radiologists usually look for to detect early signs of breast cancer: calcifications (small flecks of calcium usually clustered together) and masses (e.g. cysts or lumps) (*What Mammograms Show: Calcifications, Cysts, Fibroadenomas*, 2018). Figure 3.1 illustrates the different type of mammograms found in the dataset.

3.2.3 mini-MIAS

The "mini Mammography Image Analysis Society" dataset (mini-MIAS) is a smaller dataset of mammograms containing 322 images in greyscale Portable Gray Map (PGM) format with associated ground truth data (Suckling, 1994) and images all reduced to a uniform size of 1024 x 1024 pixels (Vishrutha and Ravishankar, 2014). The dataset contains three different types of mammograms: glandular dense, fatty and fatty glandular (see Figure 3.2), which are further divided into normal, benign and malignant cases (Hepsağ et al., 2017).

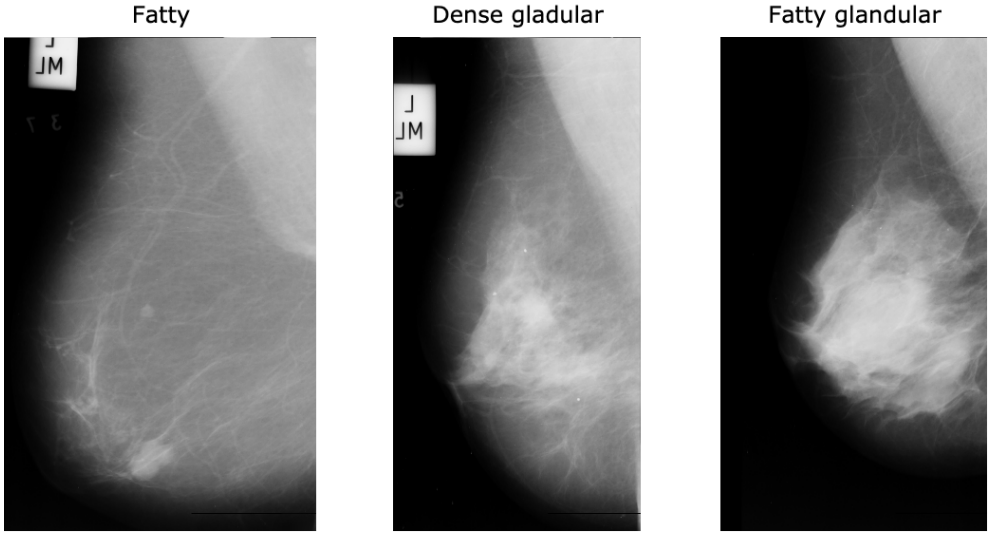


Figure 3.2: The three different types of breast background found in the mini-MIAS dataset.

Chapter 4

Design

Based on the machine learning and deep learning applications for the task of breast cancer detection established in Chapter 2, and the datasets available for this project, design decisions specific to the deep learning pipeline implementation will be covered, along with the reasoning behind the choice of datasets to use and general considerations.

4.1 Datasets Decision

An early design decision taken as a group consisted of electing which datasets to use. Despite being widely used in existing literature, popular feature-based datasets containing extracted mammogram data such as the WBCD dataset (Wolberg et al., 1995) will not be used, as the objective of using deep learning models such as CNNs is to learn which features to extract by using the raw image in 2D space rather than data flattened into 1D arrays. If extracted features such as the ones from WBCD were used, then already successful machine learning algorithms such as SVMs or DTs could be used instead of deep learning techniques.

From a clinical point of view, the mini-MIAS dataset is interesting as it contains both abnormal cases and normal cases, resulting in three classes (normal, benign and malignant cases). Its smaller size makes it useful for initial prototyping but has the downside of requiring more image processing techniques such as data augmentation to generate enough data to feed into the deep learning model. Additionally, the CBIS-DDSM dataset was chosen over the DDSM dataset as it is an updated version of the older DDSM dataset, and is curated by a trained mammographer. Additionally, it uses uncompressed images in DICOM format rather than LJPEG format, which is a deprecated format nowadays, resulting in much higher quality imagery. Indeed, the large uncompressed format offered by DICOM means that the mammograms can be fed into the CNN with larger sizes, allowing the model to potentially learn more

low-level features. Another plus is that the dataset is already split into training/testing sets, allowing for accurate performance comparisons with other papers using the same dataset.

4.2 Deep Learning Pipeline Design Analysis

The deep learning pipeline implemented for the task of breast cancer detection can be broken down in four distinct phases, which are condensed in Figure 4.1:

- **Data pre-processing**: loading a dataset in memory and processing it to gather image-label pairs for the classification task.
- **Model training**: creating a CNN model that can fit the data to learn the training set samples. Predictions are carried out once the model finishes training on the validation and test sets.
- **Result visualisation**: the model's performance is evaluated by calculating various metrics and plotting predictions.
- **Fine tuning**: a bag-of-tricks approach is used, experimenting with various deep learning techniques.

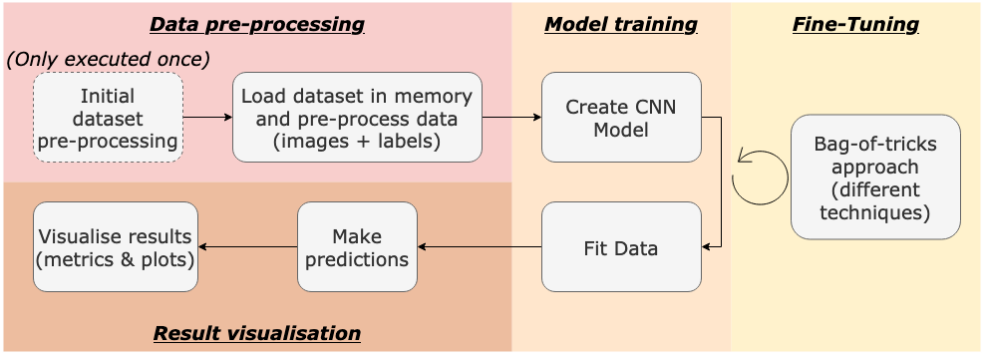
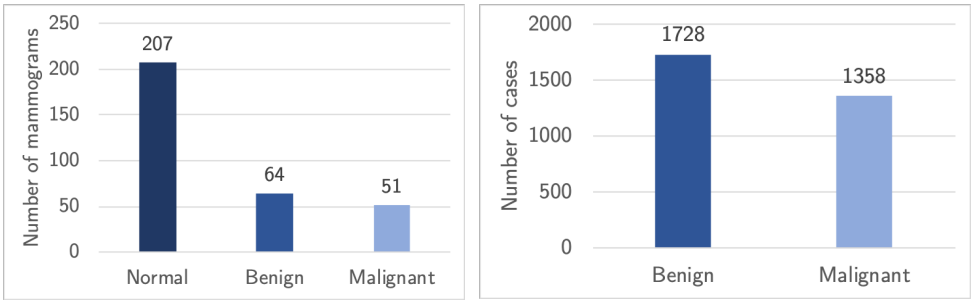


Figure 4.1: A high-level flowchart of the breast cancer detection deep learning pipeline to implement, separated into data pre-processing, model training, results visualisation and fine-tuning.

4.2.1 Data Pre-Processing

Dataset balance

As this is a classification task, it is essential to visualise the datasets' class distribution to determine whether some classes are much more frequent than other classes (skewed datasets) (Géron, 2019).



(a) mini-MIAS class distribution. (b) CBIS-DDSM class distribution.

Figure 4.2: Class distribution for the mini-MIAS and the CBIS-DDSM datasets.

The class distributions plotted in Figure 4.2 reveal that the mini-MIAS dataset is heavily imbalanced as the distribution is not uniform, which must be taken into account to avoid training a biased CNN model. Potential solutions to counter this imbalance would be to either:

- *undersample* the dataset by dropping images altogether;
- *oversample* the dataset, which can be achieved via data augmentation,
- include *class weights* to give more importance to under-represented classes.

Undersampling the dataset can be considered inefficient as it will diminish the number of samples the model could learn from by dropping samples from the majority class (Liu et al., 2009). As the datasets are already very small, undersampling them would be a poor strategy as it may discard useful features that could be learned. Consequently, oversampling by creating new artificial images resembling the original data is a viable solution as it was proven to increase accuracies. Alternatively, a cheaper option in terms of computing power that does not require the dataset to be touched would be to add class weights, which will cause the loss to become a weighted average giving more importance to less frequent classes (Zhu et al., 2018).

Dataset split

The dataset is immediately split between a training set and a testing set to avoid any form of data snooping, which corresponds to the poor practice of making design decisions (either voluntary or involuntary) after having viewed the data and detecting patterns that could lead to favouring certain models or hyperparameters above others. Due to the small size of the datasets used and to avoid causing further imbalance to the datasets, the splits are stratified to maintain representative samples from the data in both the training and the

testing sets to avoid introducing sampling bias.

An 80%/20% split, often used in machine learning and seen in breast cancer detection papers (Yue et al., 2018), is used to split the mini-MIAS dataset. The CBIS-DDSM does not need to be manually divided as it was already split with the appropriate stratification when it was designed (Lee et al., 2017). After this step, only the training set is utilised until the final results evaluations. The testing set is placed aside and forgotten about to avoid any form of cheating.

The assumption that the data never changes during the development of the project is made through a fixed random number generator seeds (see Section 5.2.2). Not using a random generator with a fixed seed would cause different samples to be extracted into the training and testing sets every time the code is executed, thus neglecting the results.

Data loading

The mini-MIAS dataset is very small in size (339 Mb before pre-processing, 202 Mb after pre-processing), containing only 322 images. It can therefore be loaded into memory without any data loading optimisation techniques. However, the CBIS-DDSM dataset is much larger, containing 10,239 images that cover 163.6 Gb of disk space. The dataset therefore cannot be loaded in memory in a single import and needs to be loaded in batches to be fed into the CNN sequentially.

Data normalisation

Images are resized to a target size during import to scale them down (CBIS-DDSM images are larger than 3000 x 5000 pixels) and to avoid having inconsistent input sizes. Observing the pixel intensities of the images found in the datasets reveal that they correspond to integers ranging from 0 to 255. However, because the weights in neural networks are small, having such large input values can disrupt and slow down the training process, ultimately leading to lower accuracies. Therefore, normalising the pixel intensities to values in the range of 0 to 1 can help fight this problem by ensuring all values are small. An example of the pixel values before and after normalisation can be seen in Figure 4.3.

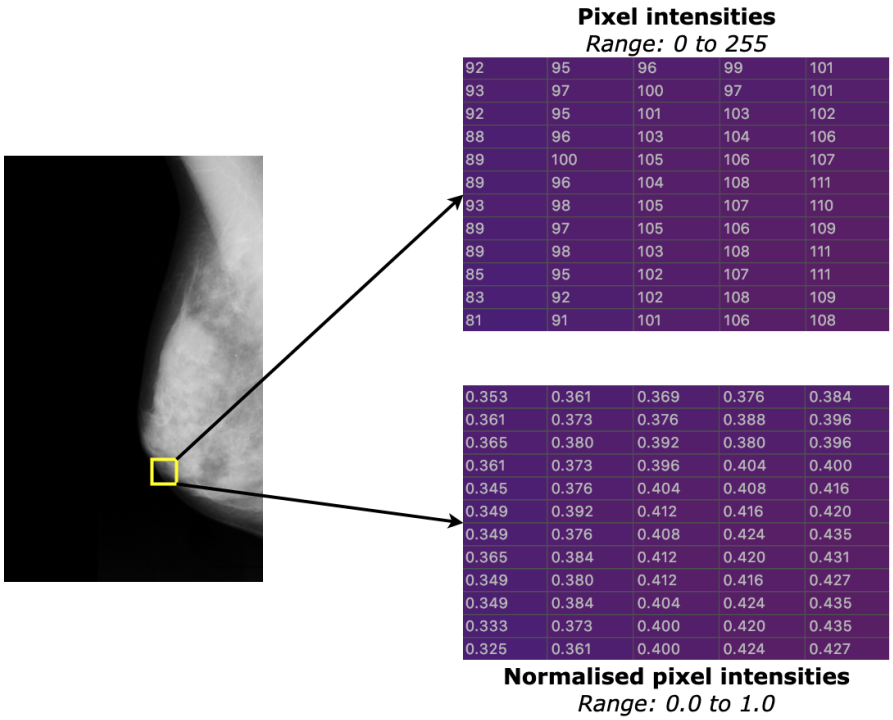


Figure 4.3: Example of the pixels values that make up a mammogram before and after normalisation.

Label encoding

As the labels for each mammogram are in categorical string format, they must be encoded into a numerical format. On the one hand, due to the sparse nature of the labels (only three categories), one-hot encoding is chosen for the mini-MIAS dataset, where a single digit may have the value 1 while the others remain at value 0 to tell apart the different labels. The one-hot encodings of the labels can be seen in Table 4.1.

Categorical format	**One-hot encoding**
Normal	1 0 0
Benign	0 1 0
Malignant	0 0 1

Table 4.1: Conversion from string format (categorical) to one-hot encoding.

On the other hand, in the case of binary datasets like CBIS-DDSM, binary encoding can be used instead of one-hot encoding, as seen in Table 4.2.

Categorical format	Binary encoding
Benign	0
Malignant	1

Table 4.2: Conversion from string format (categorical) to binary encoding.

4.2.2 Model Training

At this stage, the training data is ready to be fed into the CNN model. The classification models will learn the processed images from the training set loaded in memory before making their predictions, which will be compared with the ground truth labels for evaluation.

CNN model

Due to the small nature of the CBIS-DDSM and mini-MIAS datasets used, state-of-the-art CNN models pre-trained on large general datasets like ImageNet are used rather than creating a CNN from scratch, a technique known as transfer learning (see Section 2.3.2) that is proven to work on breast cancer detection tasks (Shen et al., 2017; Falconi et al., 2019).

Different CNN architectures can be used as the base of a custom CNN model tailored for breast cancer detection. This is achieved by using popular CNN architectures available with Keras such as VGG19, ResNet50, InceptionV3, DenseNet121 and MobileNetV2 as the base of the CNN (Keras, 2020). The fully connected layers of these models, originally designed for general classification of 1,000 different categories, are dropped from the model and replaced with a custom MLP (Krizhevsky et al., 2012). The fully-connected MLP contains hidden layers and an output layer with different activation functions based on the dataset used.

If large images are used, then additional convolutional and pooling layers are added before the base model to downsample the image to smaller sizes and learn low-level features, followed by the pre-trained base model, a flatten layer to convert the output from 2D to 1D, and finally the MLP which will make the final prediction. Dropout layers are added in the MLP to avoid overfitting. The full model can be visualised in Figure 4.4.

Data fitting

Activation functions Different activation functions can be used in the output layer of the CNN (see Figure 4.5). Typically, a single neuron with a sigmoid activation function is used for binary problems as it outputs an independent value between 0 and 1 that can be interpreted as a probability of the positive

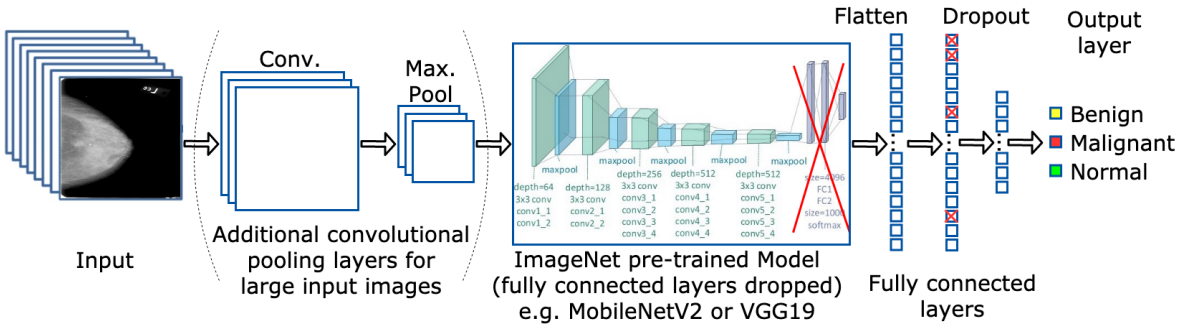


Figure 4.4: CNN architecture used. VGG19 image retrieved from `https://tinyurl.com/rpp49oc`.

class. Inversely, a softmax activation function transforms the output of the last hidden layer into probabilities for each class that sum up to 1 (Litjens et al., 2017). These probabilities are dependent of each other, as each sample must belong to one class only, making it perfect for multi-class classification of breast mammograms (each sample can only be either normal, benign or malignant, therefore to increase the probability of one class, it must decrease it for another).

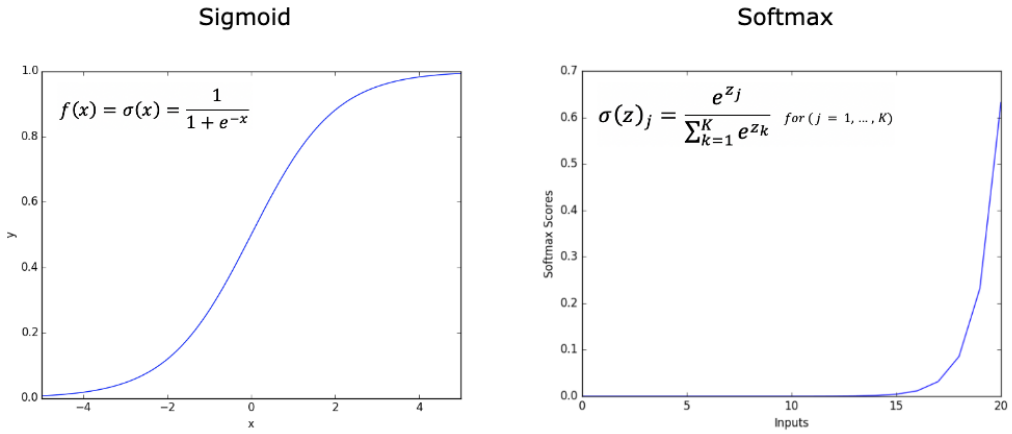


Figure 4.5: Visualisation of the sigmoid and softmax activation functions.

Resultantly, the output layer will use a sigmoid with the CBIS-DDSM dataset, and a softmax with the mini-MIAS dataset.

Loss function Cross entropy is one of the most commonly used loss functions as it can be used for both binary and multi-class tasks (Litjens et al., 2017). Because probabilities are being estimated through the sigmoid and softmax activation functions, cross entropy is the ideal loss function as it heavily

penalises the model when a low probability is predicted for the target class (Géron, 2019).

Optimiser Due to the deep nature of the model, it is important to minimise the number of hyperparameters to control. Adaptive learning rate algorithms usually generalise better than traditional optimisers like SGD or momentum, which are slow to converge and require more fine-tuning. The most general adaptive optimiser is *adaptive moment estimation* (Adam), which combines both momentum for more significant steps in the direction of the steepest gradient and Root Mean Square Prop (RMSProp) for more accelerating on steep slopes than small slopes, making it the best choice for this model.

Transfer learning training To best make use of the transfer learning technique with the base model's weights instantiated using ImageNet weights, training is separated into two phases:

1. All the layers from the base model are frozen, enabling only the custom MLP with fully connected layers to fit the mammogram images. The initial training phase ends once the maximum number of epochs is reached, or the early stopping condition is met.

2. All the layers are unfrozen, and training is resumed with a lower learning rate 0.00001, allowing the base model to slightly alter its weights to adapt to the mammogram dataset while not forgetting the ImageNet knowledge.

Validation set & early stopping To ensure that the model generalises well to unseen data from the testing set, the training set is further split to form a validation set using a 75%25% split on training set; resulting in a 60%/20%/20% of the full dataset. The validation set is used to make predictions at the end of each epoch by calculating the loss and accuracy. The validation loss is then monitored to stop the training before the maximum number of epochs is reached if the loss does not improve after a certain number of epochs, preventing the model from overfitting the data too much.

Due to time constraints, k-fold cross-validation, which divides the training set into K subsets and evaluates the model K times, cannot be used as training the model on the CBIS-DDSM dataset already takes between 1h15m-8h49 (see Chapter 6), which would be multiplied by a factor of K using cross-validation. Therefore, a validation set is used instead.

Fine-tuning Traditionally, a grid search approach would have been preferred to fine-tune the model's hyperparameters. However, due to the significant training runtime and the number of hyperparameters to fine-tune, a

grid search would have been unrealistic given the time frame of the project. Additionally, two attempts at implementing grid search were unsuccessful, as a known bug on the Keras wrapper for Scikit-Learn prevented the use of the *GridSearch class*, and Optuna was incompatible with the current version of Tensorflow I/O being used.

Instead, a bag-of-tricks approach is selected, manually trying different deep learning techniques mentioned in this chapter such as using different pre-trained CNN models, amounts of transfer learning, amounts of data augmentation, dropout values and input image sizes.

4.2.3 Result Visualisation

The following terminology is used to define the metrics used below:

- TP: True Positives (positive case correctly predicted as positive);
- TN: True Negatives (negative case correctly predicted as negative);
- FP: False Positives (negative case incorrectly predicted as positive);
- FN: False Negatives (positive case incorrectly predicted as negative).

Overall accuracy

The imbalanced class distributions (see Section 4.2.1) must be taken into account when analysing the classifiers' scores. Indeed, using an evaluation metric such as overall accuracy (see Equation 4.1, Falconi et al. (2019)) would be misleading as it would not be representative of how well the classifier fitted the data. Additionally, in breast cancer detection, detecting FPs and FNs is primordial to avoid interpreting malignant tumours as benign and vice versa, an interpretation which could harm the patient and eventually lead to their death.

$$Accuracy = \frac{TP + TN}{P + N} \tag{4.1}$$

For instance, if a dumb classifier that always classifies an image as "normal" is created, it would achieve 64.28% accuracy on the mini-MIAS dataset despite never picking up abnormal cases. Therefore, a mixture of additional metrics should be used to assess how well the model learns the mammograms data and generalises to unseen cases.

Precision & recall

Precision corresponds to the number of correct positive predictions (see Equation 4.2, Liu et al. (2009)), showing the model's ability to avoid labelling negative instances as positive.

$$Precision = \frac{TP}{TP + FP} \tag{4.2}$$

Recall is the number of positive instances that are correctly predicted (see Equation 4.3, Liu et al. (2009)), showing how well the model can find all positive instances.

$$Recall = \frac{TP}{TP + FN} \tag{4.3}$$

F1 score

Together, precision and recall can be combined into a more concise metric, the *F1 score*, which corresponds to the harmonic mean of precision and recall (see Equation 4.4, Géron (2019)). To achieve a high F1 score, both precision and recall must be high (unlike a regular mean) because as the precision goes down, the recall goes up, and vice versa, making the F1 score a reliable metric for evaluating a classifier since a balance between precision and recall must be found (Géron, 2019).

$$F_1 = \frac{2}{\frac{1}{precision} + \frac{1}{recall}} = \frac{TP}{TP + \frac{FN+FP}{2}} \tag{4.4}$$

Confusion matrix

This visual metric plots the number of predictions made for each class for each possible class in a table, with each row corresponding to the actual labels and each column corresponding to a prediction. It is beneficial for detecting which actual classes are being detected the most, and what predicted classes are being misclassified as (Bhardwaj and Tiwari, 2015; Liu et al., 2009). To further highlight the misclassifications and compare predictions with other classifiers, the confusion matrices are normalised to show a percentage rather than a count.

4.3 General Design Decisions

4.3.1 Programming Language

Multiple languages, including Python, Java, R and Javascript, are considered for this project. Ultimately, Python is chosen for this project due to the

familiarity and experience with the language; and the availability of open-source libraries for implementing common machine learning functionalities, as well as data pre-processing, manipulation and visualisation techniques found in deep learning systems to avoid manually implementing them (Raschka and Mirjalili, 2017), such as Tensorflow (Abadi et al., 2015), Keras (Chollet et al., 2015), SciKit-Learn (Pedregosa et al., 2011), Pandas (pandas development team, 2020), Matplotlib (Hunter, 2007), NumPy (Oliphant and developers, 2020) and Seaborn (Michael, 2020). Refer to Appendix B.1 for a deeper review of the main pros and cons considered.

4.3.2 Deep Learning Framework

Due to the vast nature of the datasets and complexity of the deep learning models to implement, powerful computing resources will be used in the form of Graphical Processing Units (GPU). A GeForce GTX 1060 6GB is provided by the School of Computer Science and remotely accessed via SSH to a lab machine equipped with the GPU in question running on CentOS.

To make use of the GPU's computing capabilities, deep learning frameworks with CUDA support (for parallel computing and GPU optimisations), CNN support and pre-trained models should be used. The two most popular deep learning frameworks nowadays are Tensorflow coupled with Keras, and PyTorch. Tensorflow/Keras being relatively older than PyTorch, have got more online support, which is confirmed by the number of daily downloads Keras has compared to PyTorch (ten times more), as well as the number of mentions in academic papers (see figures in Appendix B.2), making them the selected deep learning library.

4.3.3 Interface

A Command-Line Interface (CLI) is selected, allowing arguments and flags to be passed to execute different sections of the code. Arguments control the dataset to use, the CNN model, and the mode to run in (training or testing). Flags control the verbose mode to print more statements in the terminal for debugging purposes. The full set of instructions to run the code can be found in Appendix C.

4.4 Design Decisions Summary

1. Datasets:
    - CBIS-DDSM (binary classification)
    - mini-MIAS (multi-class classification)

2. Data pre-processing steps:
    - 60/20/20% train/test/validation splits
    - Data augmentation
    - Image normalisation
    - One-hot encoding (CBIS-DDSM) and binary encoding (mini-MIAS)
3. Model training:
    (a) Base model: VGG19, ResNet50, InceptionV3, DenseNet121 and MobileNetV2
    (b) Output layer activation function: softmax (CBIS-DDSM) and sigmoid (mini-MIAS)
    (c) Optimiser: Adam
    (d) Loss function: Cross entropy
    (e) Transfer learning
    (f) Generalisation to unseen data: validation loss and accuracy early stopping
    (g) Fine-tuning: bag-of-tricks approach
4. Evaluation Metrics:
    (a) Numerical metrics:
        - Overall accuracy
        - Precision & recall
        - F1 score
    (b) Visual metrics:
        - Confusion matrices (counts)
        - Normalised confusion matrices
5. Programming language:
    (a) Python 3.7
    (b) Open-source frameworks:
        - Tensorflow & Keras
        - Scikit-Learn
        - NumPy, Pandas, MatplotLib & Seaborn
6. Interface: Command-Line Interface

Chapter 5

Implementation

The implementation of the breast cancer detection system was conducted in two parts. The first part corresponds to a common pipeline developed in group (Jaamour et al., 2020*a*), and the second part to individual extensions and implementations (Jaamour et al., 2020*b*). An overview of the different pipeline parts can be found in Section 5.6, while meeting minutes with the work distribution during the implementation of the common pipeline can be found in Appendix E.

5.1 Code Design

The code, stored in the *src* directory, is designed by being split into functions spread across multiple python modules, which are all organised into different directories based on the kind of task they are designed to carry out (see Appendix F).

Flow of execution The program entry point is in the "main.py" module, which parses command-line arguments used to execute different parts of the program (e.g. which dataset to use) and stores them in "config.py". The main flow of the pipeline is controlled in "main.py", enclosing calls to different functions for processing the data, creating the CNN models and evaluating the results based on the CLI arguments selected.

Data pre-processing Functions linked to data pre-processing, such as retrieving image paths and labels, processing images, encoding labels, loading the data into memory and applying transformations for data augmentation, are all located in the *data_operations* directory. The one-time scripts for initially parsing the images' paths and labels for each dataset are in the *dataset_processing_scripts* directory.

CNN Model All CNN-related code, from creating the Keras model to

compiling it, fitting it, making predictions or evaluating it is placed in a custom *CNN_Model* class, which can be found in the *cnn_models.py* module. Each model is then placed in individual modules in the *cnn_models* directory.

Results visualisation Functions handling result visualisations in the form of plots and CSV reports are all situated in the *data_visualisation* directory. Each figure and CSV report is saved in an *output* directory, while model weights are stored on an external filesystem (BigTMP) due to their large size.

Other General functions for printing information in the terminal and common operations are all located in the *utils.py* module, while global command-line arguments used throughout the code are all stored in the *config.py* module.

5.2 General

5.2.1 Command-Line Interface

Python's *argparse* module is used to implement the CLI application, accepting different arguments (e.g. dataset, type of mammogram, CNN base model, learning rate, batch size, maximum number of epochs, etc.) that are used across the pipeline to execute different sections of the code or use different hyperparameters (see Appendix C.2). The values passed through the command-line are then stored as variables in the *config.py* module.

5.2.2 Results reproducibility

To reproduce results across different runs, random number generators are seeded with an identical value. The NumPy and Tensorflow seeds are both set to 111, as well as functions that include randomness such as the dataset splits to ensure that constant shuffle indices are used (Scikit-Learn's *train_test_split*) and the dropout layer to ensure that the same random neurons are dropped:

```
# Random number generators.
numpy.random.seed(config.RANDOM_SEED)  # NumPy
tf.random.set_seed(config.RANDOM_SEED)  # Tensorflow

# Dataset splits.
_, _, _, _ = train_test_split(dataset, labels, test_size=0.25,
    stratify=labels, random_state=config.RANDOM_SEED, shuffle=True)

# Dropout layer.
fully_connected.add(Dropout(0.2, seed=config.RANDOM_SEED))
```

5.3 Data Pre-Processing

5.3.1 Initial Dataset Processing

When downloaded, both datasets contain nested directories with the mammogram images and CSV files mapping images to their label, along with additional unrequired information (e.g. breast densities or mass shape). Two distinct Python scripts are written to parse these CSV files.

mini-MIAS The data is first cleaned by replacing empty cells with the "N" character for normal cases (only benign and malignant cases are specified initially), and each image is converted from PGM to PNG format before being saved in labelled directories rather than one vast directory.

CBIS-DDSM Mass and calcification mammograms are grouped into the same set by creating a new CSV file for training and testing data containing the image name, its path and its label. The CBIS-DDSM dataset is not stored locally as its size exceeds 160GB, and is therefore stored on *BigTMP*, a 15TB filesystem that is mounted on the Centos 7 computer lab clients.

5.3.2 Data Loading

The data is loaded by parsing the generated CSV files mentioned above, loading mammogram-label pairs simultaneously. In the case of the CBIS-DDSM dataset, a Tensorflow *Dataset* is used to handle its large size by loading consecutive images into batches and caching the loaded data. This process is optimised by importing the data with parallel optimisations through Keras' *prefetch method* and the *data.experimental.AUTOTUNE* option.

5.3.3 Data Processing

Image processing The mini-MIAS mammograms are first imported through Keras' *load_img* function in Python Imaging Library (PIL) format and resized to a target size (e.g. 224 x 224 pixels) in greyscale (single channel) before being converted to array format using the *img_to_array* function to be compatible with the Keras CNN models expected input. Additionally, if the *roi* flag is *True*, then the images are cropped around a 224 x 224 ROI (Region of Interest) in the presence of an abnormality; otherwise around a 224 x 224 region in the centre of the image.

The images are then normalised by dividing their pixel intensities by 255. Because the CBIS-DDSM mammograms are in DICOM format, they first need to be imported as raw bytes before being decoded using Tensorflow I/O's

decode_dicom_image function. The image is then converted to PNG before following the same process as the mini-MIAS images.

Label encoding In the case of multi-class classification, Scikit-Learn's *LabelEncoder* class is used to one-hot encode labels, whereas Keras' *to_categorical* function is used to convert labels to a binary class matrix in the case of binary classification.

5.3.4 Dataset Splits

Scikit-Learn's *train_test_split* function is used to split the dataset into training, validation and testing sets with 60/20/20% shuffled stratified splits (see Figure 5.1).

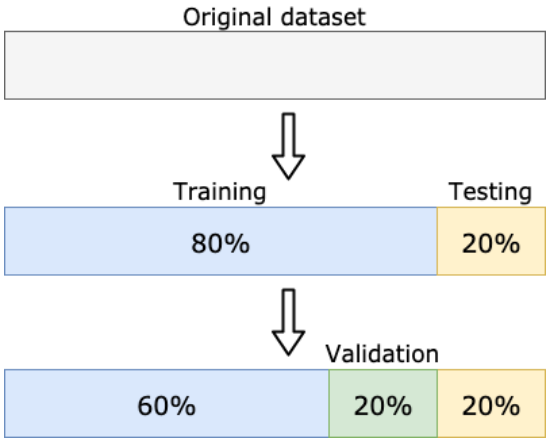


Figure 5.1: Original dataset divided into training, validation and testing sets using a 60/20/20% split.

5.3.5 Data Augmentation & Class Balance

For the highly imbalanced mini-MIAS dataset, image augmentation is used to balance the class distribution. Random amounts of rotations and shears between -20 and 20 degrees, noise and horizontal flips are added through the Scikit-Image library to existing images of the minority classes to balance the dataset. An integer variable is used to control the augmentation factor. These are then shuffled to ensure that newly generated images are not grouped together. An example of the possible transformations is shown in Figure 5.2.

A computationally-cheaper alternative consists of calculating class weights to balance the classes, giving larger weights to the minority class and smaller weights to the majority class. For example, in the case of the CBIS-DDSM

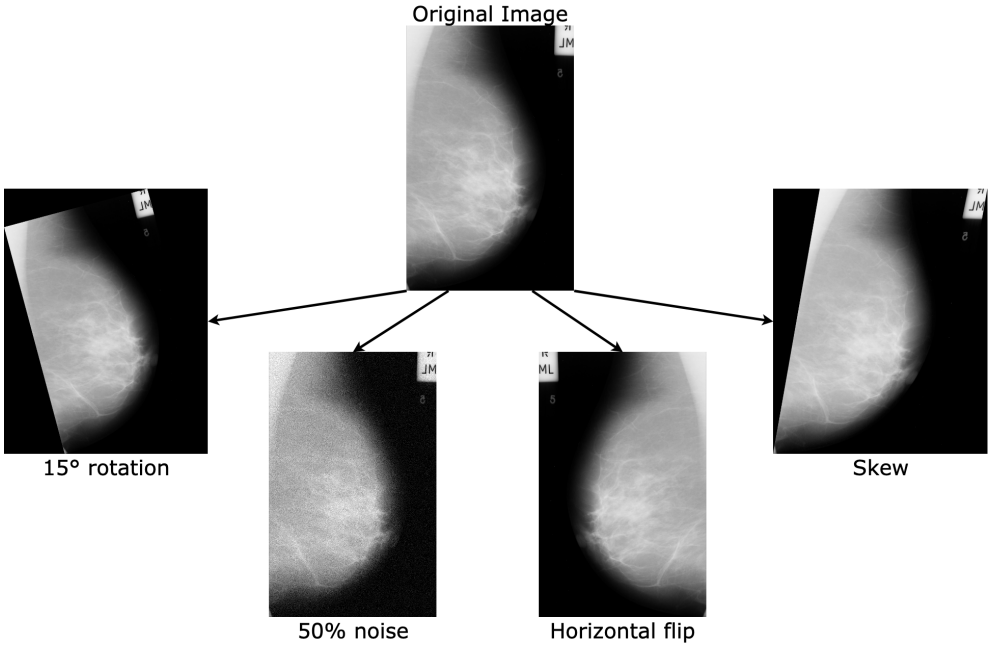


Figure 5.2: Example of affine transforms applied to mini-MIAS mammograms to generate new samples. Original image retrieved from the mini-MIAS dataset (Suckling, 1994).

dataset (see Section 4.2.1), a weight of 0.907 can be used for benign samples and 1.113 for malignant samples to balance the dataset. These weights are calculated by using Scikit-Learn's *compute_class_weights* function with the "balanced" argument.

5.4 Model Training

5.4.1 Sequential Model

The CNN model is created in Keras through the custom *Cnn_model* class. The Keras *Sequential* class is used to create the CNN model described in Section 4.2.2 by linearly stacking layers on top of one another. The input layer size is first specified to match the target size chosen in the pre-processing steps. Because a CNN pre-trained on ImageNet with RGB images (three channels) is used, the greyscale images have to be concatenated into a triple channel greyscale input:

```
img_input = Input(shape=(config.IMG_SIZE['HEIGHT'], config.
    IMG_SIZE['WIDTH'], 1))
img_conc = Concatenate()([img_input, img_input, img_input])
```

Next, the CNN architecture pre-trained on ImageNet is used through the

Keras Application API, allowing any CNN to be used such as VGG19 or MobileNetV2. The fully connected layers are dropped by adding *include_top=False* to the imported model. To download the ImageNet weights automatically, the following line of code is included to bypass SSL certificate verifications:

```
ssl._create_default_https_context = ssl._create_unverified_context
model_base = MobileNetV2(include_top=False, weights="imagenet",
    input_tensor=img_conc)
```

Next, a *Flatten* layer is added to convert the data from 2D to 1D, making it compatible with a fully-connected MLP. In this MLP, a dropout layer with a rate of 0.2 is added, followed by two dense hidden layers (one with 512 neurons, the other with 32 neurons). Finally, the output layer is appended at the end of the sequential model, with either a sigmoid activation function if the CBIS-DDSM dataset is used or a softmax activation function if the mini-MIAS dataset is used. If large images are used (e.g. 1024 x 1024 pixels or larger), then additional convolution and pooling layers are added before the pre-trained model using Keras' *Conv2D* and *MaxPooling2D* classes.

5.4.2 Training Steps

The training phase is separated in two steps, as mentioned in Section 4.2.2. To freeze the pre-trained base model's layers, the *trainable* attribute is set to *False*, allowing only the MLP's fully connected layers to learn. During this step, the learning rate and the maximum number of epochs are set by the CLI arguments, which default to 0.001 and 150 respectively. Once training converges, all layers are unfrozen by setting *trainable* attribute to *True* and a second training phase is launched with a lower learning rate set to 0.00001.

Tensorflow *Callbacks* are used to enable early stopping conditions, which are set to halt training when the validation loss does not improve after the maximum number epochs specified divided by 10 (e.g. 150/10 = 15). Additionally, an option to reduce the learning rate on plateaus is used to avoid increasing the validation loss when the learning stagnates, which is set at half the previously mentioned patience.

For the CBIS-DDSM dataset, the *BinaryCrossentropy* loss and *BinaryAccuracy* are used, whereas *CategoricalCrossentropy* and *CategoricalAccuracy* are used for the mini-MIAS dataset. This is necessary as different forms of cross entropy and accuracy equations have to be used to accommodate the binary classification tasks which have labels encoded differently (binary-encoded labels for CBIS-DDSM and one-hot-encoded labels for mini-MIAS).

5.4.3 Model & Weights Saving

Once the CNN is finished training, the full model, including the model's configuration, the layer hyperparameters, the optimiser, the training results (losses and metrics), and the model's state (layer weights) are saved in HDF5 format (".h5" file extension) to be re-used for the model's final evaluation on the test set. Additionally, specific layer weights are saved for transfer learning experiments, such as the entire model's weights or the fully connected layers' weights only, to test different levels of transfer learning (see Section 6.7).

To perform transfer learning from the mini-MIAS dataset to the CBIS-DDSM dataset, the mini-MIAS dataset is binarised by dropping normal cases altogether, creating a tiny dataset of 115 abnormal images (64 benign and 51 malignant). The model is then trained on the binary mini-MIAS dataset and its final weights are saved in NumPy format. These weights can be loaded when instantiating a new model for fitting the CBIS-DDSM dataset.

5.5 Predictions & Results visualisation

Once the model is trained, predictions can be made through the *make_prediction* function by passing in new images and their ground truth labels. These predictions are then evaluated through the metrics mentioned in Section 4.2.3. The accuracy, precision, recall, F1 score and confusion matrices are all calculated via Scikit-Learn's *metrics* API. Additionally, the evolution of the loss and accuracy on the training and validation sets are all plotted against the number of epochs during both phases of training to assess how well the model learned the data and where improvements could be made. The results above are either saved in CSV files or plotted using Matplotlib and Seaborn before being saved in PNG format to later be referenced in Chapter 6.

During the development of this pipeline, the validation set was used to assess the performance of the model without snooping into the test dataset, which was reserved for the final evaluation of the model in Chapter 6.

5.6 Pipeline Flowchart

The flowchart in Figure 5.3 reveals which sections of the pipeline were implemented as a group, later amended, and implemented individually.

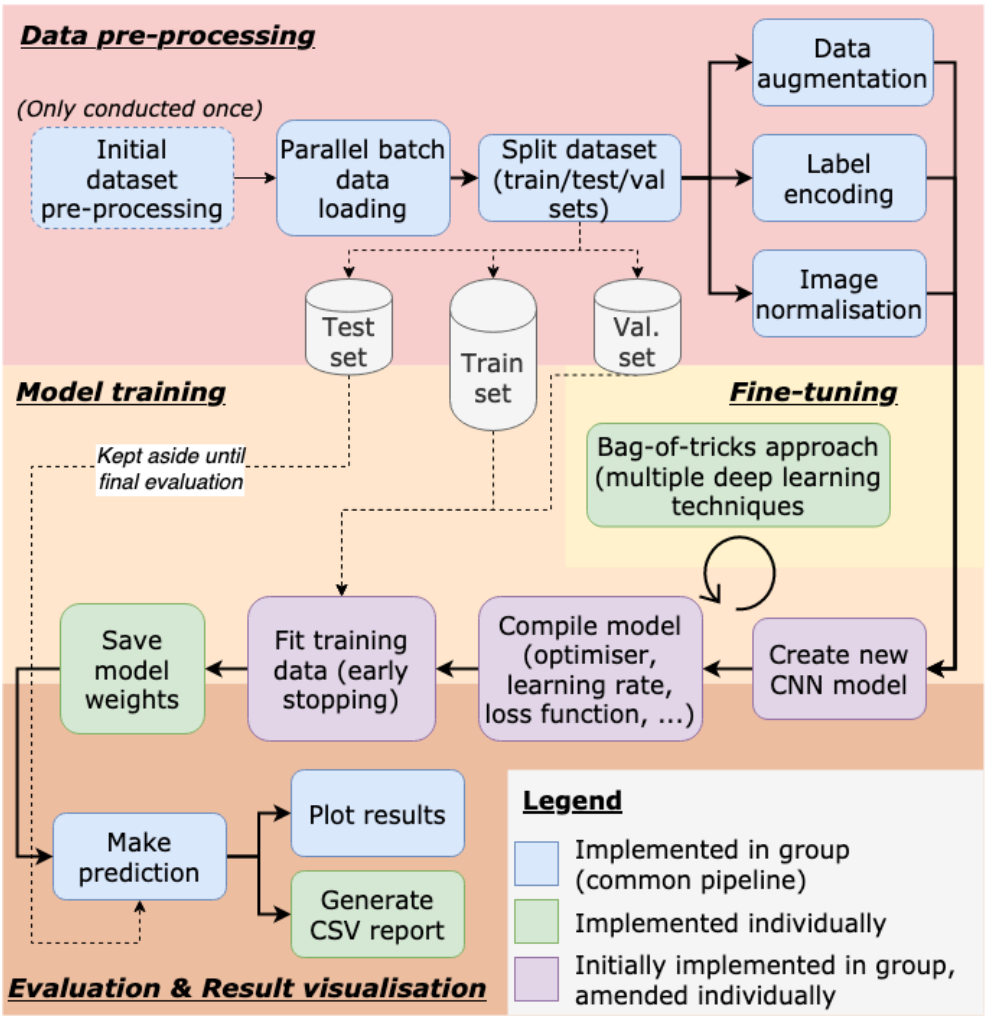


Figure 5.3: A detailed flowchart of the breast cancer detection deep learning pipeline implemented, separated between data pre-processing, model training, results visualisation and fine-tuning.

Chapter 6

Results & Evaluation

This section covers the bag-of-tricks approach mentioned in Section 4.2.2, where multiple deep learning techniques covered throughout Chapters 2 & 4 are experimented with to determine which improve the performance of the model. Across each experiment, identical configurations are used to ensure that accurate comparisons can be made.

6.1 Test Data

CBIS-DDSM dataset contains the following number of test samples:

- Total: 641
  - Benign: 381
  - Malignant: 260

The mini-MIAS dataset contains the following number of test samples:

- Total: 65
  - Normal: 42
  - Benign: 13
  - Malignant: 10

6.2 Model Used

The model described in Section 4.2.2 is used across all the experiments in this chapter. Only the dataset, base CNN architecture, batch size, class weights, data augmentation factor, input size, weight initialisation and type of mammograms vary across the following experiments. The following remain constant across the experiments:

- fully connected MLP with 512 and 32 hidden neurons and 2/3 output neurons;

- dropout layer using $p = 0.2$;
- Adam optimiser with a learning rate of 0.001 for VGG19 and 0.0001 for MobileNetV2;
- whole images.

6.3 Baseline Results

An overall accuracy of 63.96% is achieved using the deep learning pipeline developed as a group (Jaamour et al., 2020*a*), and is used as a benchmark to compare the results obtained through the different bag-of-tricks techniques.

6.4 Base CNN Architectures

Five different CNN model architectures pre-trained on ImageNet (VGG19, ResNet50, InceptionV3, DenseNet121 and MobileNetV2) are tested out as the model's base. For this test, the CBIS-DDSM dataset is used with whole images resized to 512 x 512 pixels, a batch size of 2 and a learning rate of 0.0001.

CNN Architecture	Overall Accuracy	Precision	Recall	F1 Score
VGG19	63.96%	63.73%	63.96%	63.83%
ResNet50	61.00%	64.23%	61.00%	61.23%
InceptionV3	62.71%	62.52%	62.71%	62.60%
DenseNet121	64.74%	65.81%	64.74%	65.03%
MobileNetV2	**66.46%**	**66.25%**	**66.46%**	**66.33%**

Table 6.1: Results achieved on the CBIS-DDSm test set when using different CNN architectures as the base model pre-trained on ImageNet weights.

The results found in Table 6.1 clearly reveal that MobileNetV2 unlocks more performance than the other CNN architectures with a higher accuracy and F1 score. The original VGG19 architecture used during the common pipeline development is outperformed by more efficient models like DenseNet121 or MobileNetV2 but outperforms ResNet50 and InceptionV3. These results contradict Falconi's results on the CBIS-DDSM dataset, who finds that ResNet50 outperforms MobileNetV2 (Falconi et al., 2019). However, MobileNetV2 still outperforms InceptionV3. These results may differ due to the different pre-processing techniques being used as Falconi uses cropped images around ROIs, whereas whole images are used in this experiment.

It is also worth noting that using MobileNetV2 a base architecture (66.46% accuracy) already surpasses the baseline (63.96% accuracy), as well as models

that use traditional machine learning methods like SVMs with GLCM features (63.03% accuracy) on the CBIS-DDSM dataset (Sarosa et al., 2018), confirming the points made in Section 2.4.

However, observing the training and testing runtimes in Figure 6.1 reveals that VGG19 takes the longest time to train with 3h50m, whereas the more efficient MobileNetV2 architecture takes 2h46m. Additionally, prediction runtime is 2.3 times faster with MobileNetV2 compared to VGG19, which is more useful for clinics as mammogram diagnosis results can be returned faster.

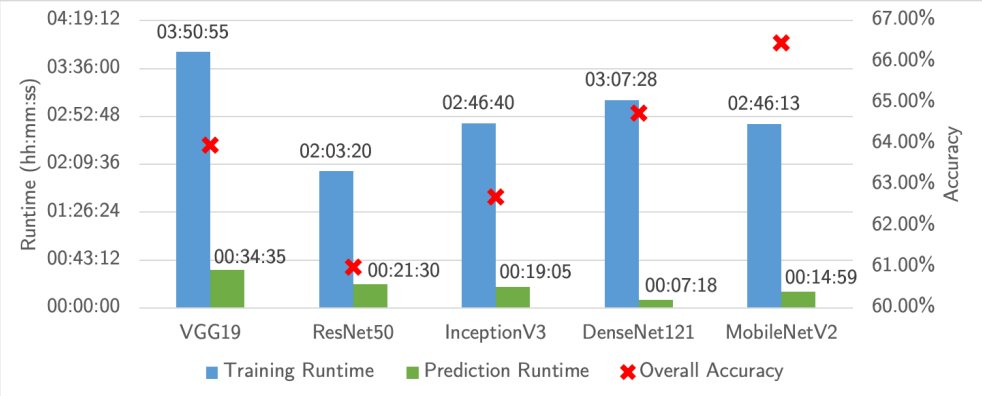


Figure 6.1: Training (2445 samples) and prediction (641 samples) runtimes on the CBIS-DDSM dataset when using different CNN architectures as the base model pre-trained on ImageNet .

6.5 Class Imbalance

6.5.1 Data Augmentation

Varying amounts of data augmentation are carried out on the mini-MIAS dataset (due to time constraints, this experiment could not be tested on the CBIS-DDSM dataset):

- No data augmentation;
- Data augmentation to balance class (create artificial benign and malignant samples to reach the number of normal samples);
- Double data augmentation.

Table 6.2 betrays the inefficiency of data augmentation on small datasets like mini-MIAS. Even doubling the amount of data maintains the same accuracy (64.62%) as no data augmentation, confirmed by the identical accuracy and recall, which indicates that the model is overfitting. Indeed, the confusion

Base Model	Data Augmentation	Overall Accuracy	Precision	Recall	F1 Score
VGG19	***None***	**64.62%**	**41.75%**	**64.62%**	**50.73%**
	Class Balance	36.92%	48.89%	36.92%	40.45%
	Double	**64.62%**	**41.75%**	**64.62%**	**50.73%**
MobileNetV2	***None***	**64.62%**	**41.75%**	**64.62%**	**50.73%**
	Class Balance	41.54%	40.70%	41.54%	40.96%
	Double	**64.62%**	**41.75%**	**64.62%**	**50.73%**

Table 6.2: Results achieved on the mini-MIAS test set when using different amounts of data augmentation (none, balanced and double).

matrix in Figure 6.2 confirms that all test samples are classified as normal despite the data augmentation.

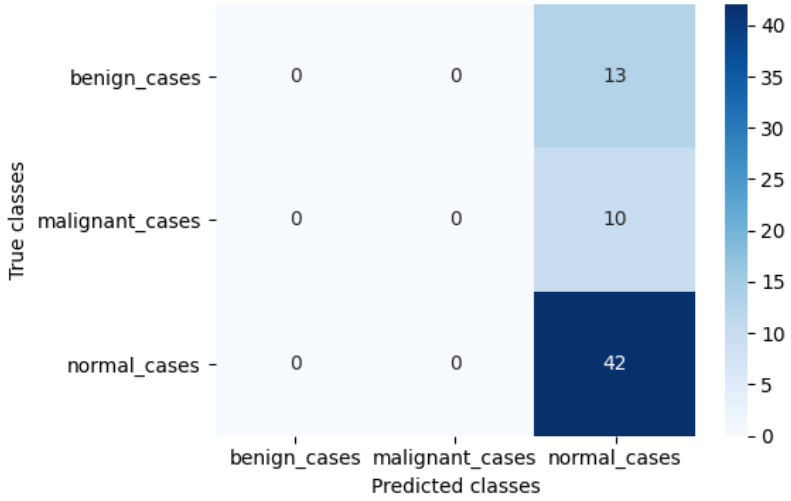


Figure 6.2: Confusion matrix when double data augmentation is applied on the mini-MIAS test dataset with MobileNetV2 as the base model.

Only when a few artificial samples are created for minority classes can the model make predictions other than "normal", but not well enough as the accuracy tumbles to 36.92-41.54% depending on the base model used. This model performs worse than other papers on the mini-MIAS dataset, which can be due to the lack of image pre-processing used as Hepsag crops the images around the ROI rather than using whole images, achieving 68% accuracy (Hepsağ et al., 2017).

In terms of the training runtime witnessed in Figure 6.3, the more data augmentation is applied, the longer the runtime is, which is expected as there is more data to process.

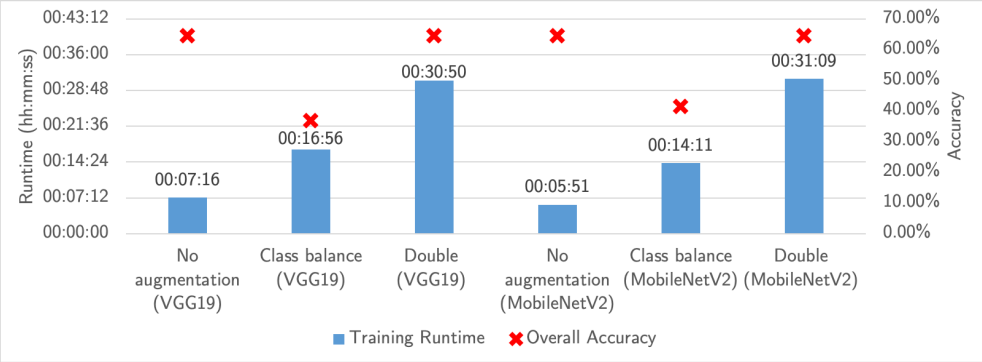


Figure 6.3: Training runtimes when using no data augmentation, augmentation to fix class balance and to double the training set size (744, 372 and 192 samples respectively) on the mini-MIAS dataset.

6.5.2 Class Weights

Distinct variations of class weights are used on the CBIS-DDSM dataset to attempt to rectify the adverse effects that can be introduced by imbalanced datasets without going through the unsuccessful process of data augmentation, which considerably slows down the training time. Table 6.3 reports the three class weight values that were tested using the imbalanced CBIS-DDSM dataset with whole images resized to 512 x 512 pixels, a batch size of 2 and a learning rate of 0.0001:

- No class weights (dataset remains imbalanced);
- Balanced class weights:
  - 0.907 for majority class (benign),
  - 1.113 for minority class (malignant);
- +50% class weight for minority class:
  - 1.0 for benign samples,
  - 1.5 for malignant samples.

These results clearly depict how including balanced weights to the samples increases the accuracy across different base CNN models by 1.25-1.71%, thus helping against the imbalanced dataset issue without resorting to techniques like data augmentation. However, a manual weight increase for the minority class decreases the accuracy by 0.78-1.1%, revealing the complexity of finding the right parameters for balancing datasets as the 50% weight increase for malignant samples made the dataset even more imbalanced. The normalised confusion matrices found in Figures 6.4 and 6.5 expose how including class

Base model	Class weights	Overall Accuracy	Precision	Recall	F1 Score
VGG19	***None***	62.25%	63.66%	62.25%	62.58%
	Balanced	63.96%	63.94%	63.96%	63.95%
	+50% minority	61.15%	62.16%	61.15%	61.45%
MobileNetV2	***None***	65.83%	66.33%	65.83%	66.01%
	Balanced	**67.08%**	**66.50%**	**67.08%**	**66.48%**
	+50% minority	65.05%	65.19%	65.05%	65.12%

Table 6.3: Results achieved on the CBIS-DDSM test set when using different class weights (none, balanced and +50% minority class) with VGG19 and MobileNet architectures as base model.

weights leads to the model being more confused as many malignant samples are classified as benign.

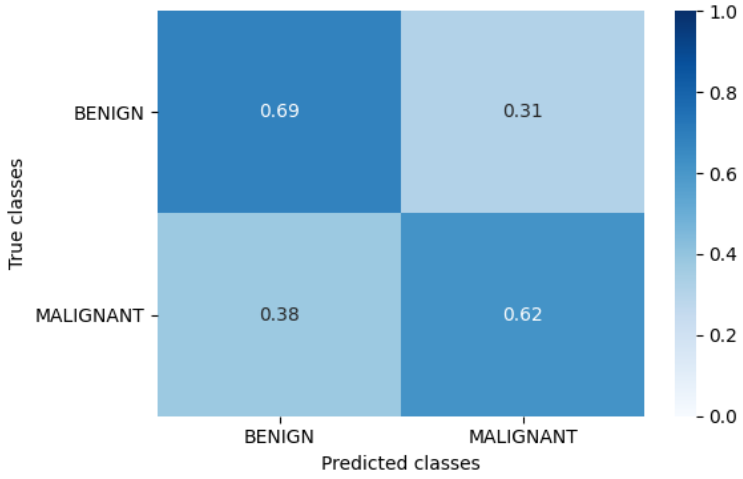


Figure 6.4: Normalised confusion matrix when no class weights are used with MobileNetV2 as the base model on the CBIS-DDSM dataset.

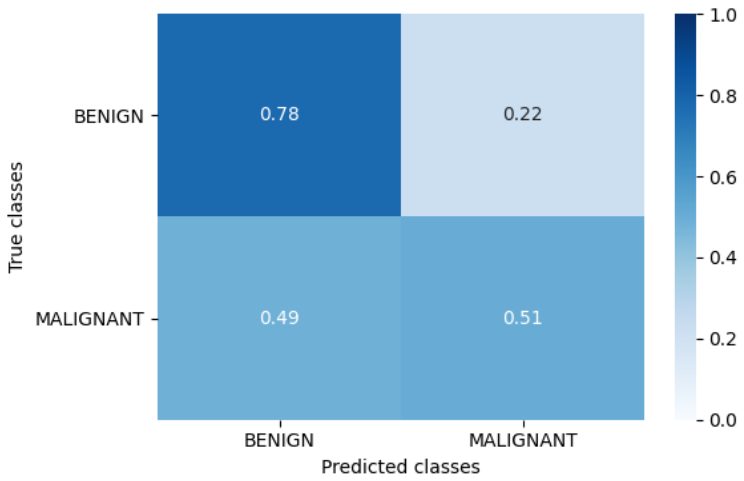


Figure 6.5: Normalised confusion matrix when balanced class weights are used with MobileNetV2 as the base model on the CBIS-DDSM dataset.

6.6 Input Image Size

Different image sizes are explored to determine their effect on the model's performance on the CBIS-DDSM dataset. For the smaller image sizes, larger batch sizes are used, whereas for the larger image sizes, smaller batch numbers are defined along with the extra convolutional and pooling layers mentioned in Section 5.4.1 to accommodate the larger image size:

- 224 x 224 pixels (chosen as most CNNs pre-trained on ImageNet use this size) with a batch size of 8;
- 512 x 512 pixels with a batch size of 2;
- 1024 x 1024 pixels (with additional convolutional/pooling layers) with a batch size of 2.

The results in Table 6.4 clearly expose the accuracy increase when using 512 pixels-wide input size rather than 224, with a 0.63% increase on VGG19 and 4.52% increase on MobileNetV2. However, further increasing the input size to 1024 pixels has no positive effect as the accuracy drops by 4.52% on VGG19 and leads to an Out Of Memory (OOM) error on MobileNetV2, despite lowering the batch size to 1.

Base Model	Whole Image Size (pixels)	Extra conv/ pool layers	Overall Accuracy	Precision	Recall	F1 Score
VGG19	***224 x 224***	No	63.96%	65.33%	63.96%	64.28%
	512 x 512	No	64.59%	64.44%	64.59%	64.51%
	1024 x 1024	Yes	59.28%	**66.94%**	59.28%	58.43%
MobileNetV2	***224 x 224***	No	62.56%	62.38%	62.56%	62.46%
	512 x 512	No	**67.08%**	66.50%	**67.08%**	**66.48%**
	1024 x 1024	Yes	OOM	OOM	OOM	OOM

Table 6.4: Results achieved on the CBIS-DDSM test set when using different input image sizes (224, 512 and 1024 pixels).

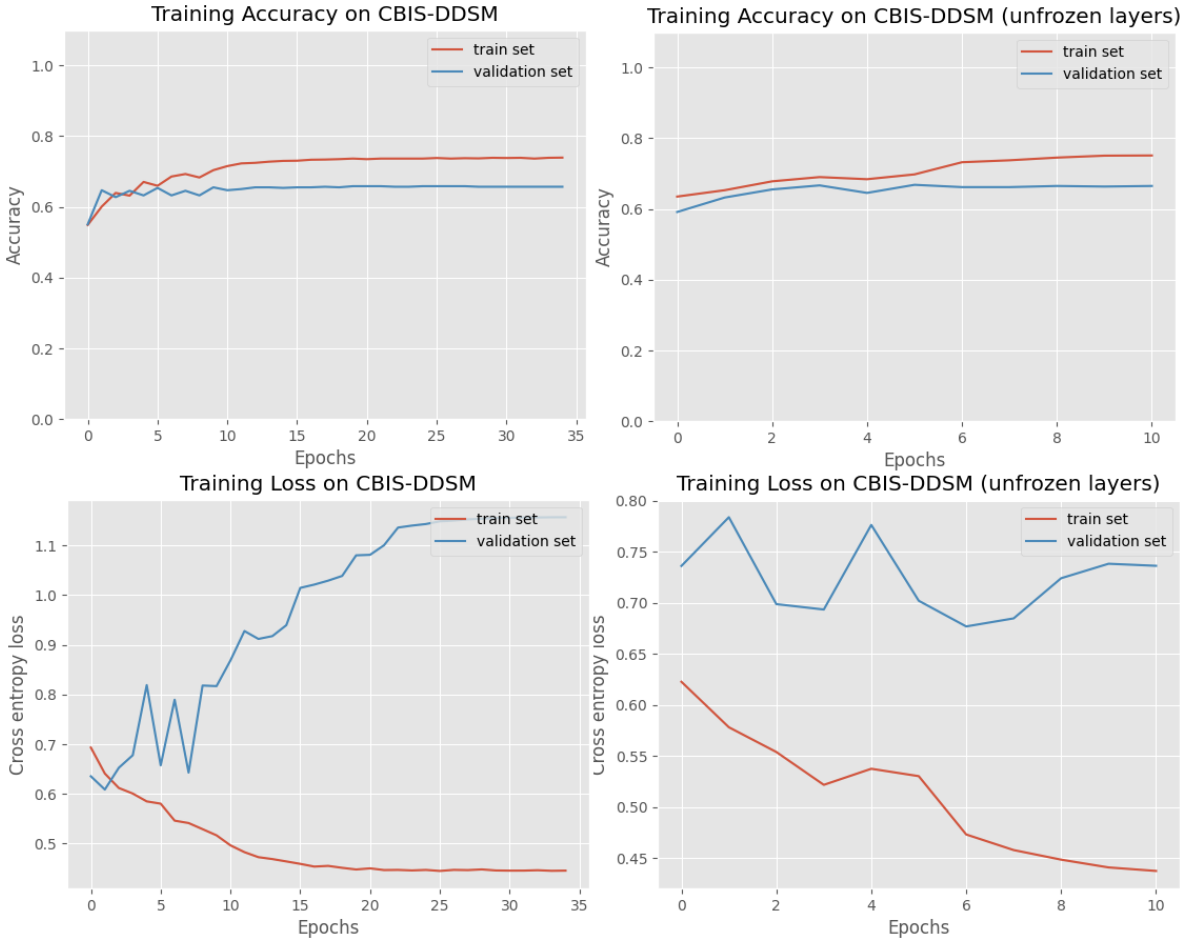


Figure 6.6: Evolution of the accuracy and loss during both training phases when testing 1024x1024 input size on VGG19.

Observing the evolution of the training accuracy and loss when using 1024 x 1024 pixels input size on VGG19 (see Figure 6.6), it can be seen that the validation loss increases while the training loss decreases and that both sets' training accuracies are increasing as well; which is a typical pattern of a model overfitting the data. Because the model is overfitting the data, a very high precision (66.94%) but low recall (59.28%) is witnessed in Table 6.4 for 1024x1024 input size, which is hugely detrimental as a BCD system that detects malignant cases as benign could lead to the death of the patient.

As expected, increasing the image size also increases the training runtime (see Figure 6.7), which is boosted by a factor of 2.4 when increasing from 224 to 512 pixels, and a factor of 2.8 from 512 to 1024 pixels on VGG19. However, another advantage of MobileNetV2 over VGG19 is that it scales better to larger input sizes as increasing the input from 224 to 512 pixels only raises the runtime by a factor of 1.54, and prediction times are quicker than VGG19 predictions (13.5 minutes on average for MobileNetV2 compared to 21.3 minutes for VGG19).

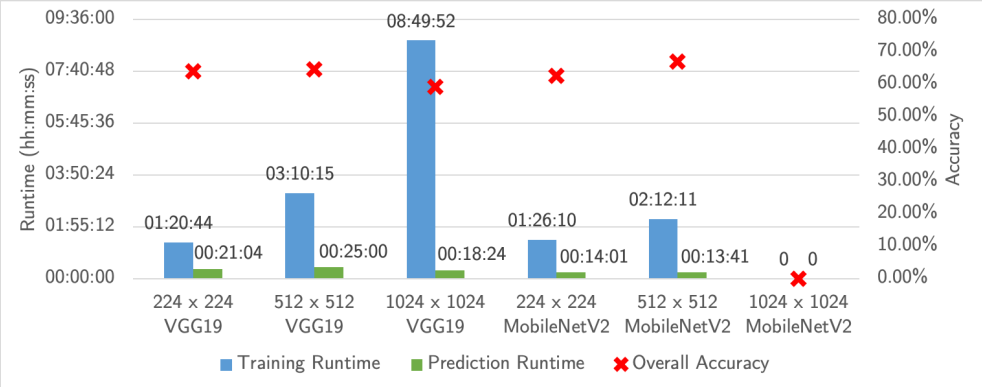


Figure 6.7: Training (2445 samples) and prediction (641 samples) runtimes on the CBIS-DDSM dataset when using different input image sizes (224, 512 and 1024 pixels).

Nevertheless, the accuracy/training runtime trade-off is not primordial in breast cancer detection as the primary goal is to develop a system that can correctly diagnose early forms of cancers in mammograms as accurately as possible, regardless of the runtime. Ultimately, prediction runtimes will matter when used in clinics.

6.7 Varying Amounts of Transfer Learning

This experiment consists of expanding upon the concept of transfer learning. Instead of using a CNN pre-trained on ImageNet, weights of a model trained on

a *binarised* mini-MIAS dataset are transferred to the CBIS-DDSM dataset (see Section 5.4.3). Four different experiments using an identical CNN architecture are tested to assess the effect of transfer learning from the binarised mini-MIAS dataset and ImageNet to the CBIS-DDSM dataset:

- Transfer learning of all layer weights (MobileNetV2 and MLP layers instantiated with binary mini-MIAS weights);
- Transfer learning of fully connected layer weights (MLP layers instantiated with binary mini-MIAS weights, MobileNetV2 layers instantiated with ImageNet weights);
- Transfer learning of ImageNet weights only (MLP layers instantiated with random weights, MobileNetV2 layers instantiated with ImageNet weights);
- No transfer learning (MobileNetV2 and MLP connected layers instantiated with random weights).

Transfer Learning	**Overall Accuracy**	**Precision**	**Recall**	**F1 Score**
Full mini-MIAS-binary transfer learning	63.18%	64.07%	63.18%	63.45%
MLP layers mini-MIAS-binary transfer learning	**67.08%**	**67.28%**	**67.08%**	**67.17%**
MobileNet layers ImageNet transfer learning	67.08%	66.50%	67.08%	66.48%
No transfer learning (random weights)	61.62%	61.00%	61.62%	61.17%

Table 6.5: Results achieved on the test set when using different amounts of transfer learning on the CBIS-DDSM dataset with the MobileNetV2 base model.

The results in Table 6.5 clearly indicate that any form of transfer learning is better than random weight initialisation with such a small dataset. On the other hand, too much transfer learning by using all the weights from the model trained on the binary mini-MIAS dataset does not generalise well to the CBIS-DDSM dataset. The best performance came from initialising MobileNetV2 with ImageNet weights and the MLP layers with the MLP layer weights trained on the binary mini-MIAS, achieving an F1 score of 67.17%. This was closely followed by again using ImageNet weights for MobileNetV2

and random weights for the MLP layers which reached the same overall accuracy but a lower F1 score (66.48%).

The ImageNet weights transfer confirmed the performance that can be gained, as well the adaptive nature of CNNs when using knowledge learned from large general datasets for a more specific task like breast cancer detection.

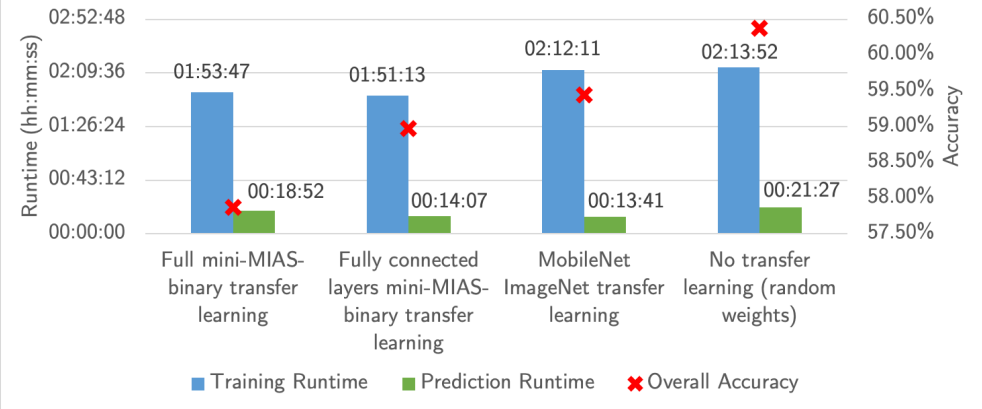


Figure 6.8: Training (2445 samples) and prediction (641 samples) runtimes on the CBIS-DDSM dataset when using different amounts of transfer learning through the binary mini-MIAS and ImageNet datasets with MobileNetV2 as a base model.

It is worth noting that training is slightly quicker when using weights from binary mini-MIAS (see Figure 6.8) as the model converges more quickly towards a known solution. However, it is the early stopping conditions mentioned in Section 4.2.2 that dictate the training duration.

6.8 Mammogram Types

To assess how the model would adapt to samples from a single mammogram type, the CBIS-DDSM dataset was separated into only masses samples and only calcifications samples (see Section 3.2.2 for samples). Three different experiments were tested:

- All types of mammograms (masses + calcifications);
- Mass mammograms only;
- Calcification mammograms only.

These results show that the model using VGG19 as a base model learns the data much better when masses and calcifications are separated, reaching

Base Model	Mammogram type	Overall Accuracy	Precision	Recall	F1 Score
VGG19	***All***	59.44%	35.33%	59.44%	44.32%
	Masses	64.35%	63.70%	64.35%	63.86%
	Calcifications	**66.67%**	**67.05%**	**66.67%**	**66.80%**
MobileNetV2	***All***	**67.08%**	**66.50%**	**67.08%**	**66.48%**
	Masses	63.23%	63.23%	63.23%	63.23%
	Calcifications	63.12%	64.57%	63.12%	63.39%

Table 6.6: Results achieved with on the test set when using different types of mammograms (VGG19, ResNet50, InceptionV3, DenseNet121 and MobileNetV2) on the CBIS-DDSM dataset.

64.35% and 66.67% accuracy respectively on the test set, but only managing 59.44% when using the full CBIS-DDSM dataset. Indeed, the normalised confusion matrix on the full CBIS-DDSM dataset indicates that all instances are classified as "benign", indicating that the model gets confused when dealing with multiple views and cannot tell benign and malignant cases from each other. This outcome is in line with Hepsag's results, which achieve higher accuracies when classifying either masses or calcifications on another dataset (Hepsağ et al., 2017), and confirms Elter's claim that masses are harder to detect than calcifications (Elter and Horsch, 2009).

However, the opposite effect is witnessed when MobileNetV2 is used as a base model, reaching an accuracy of 67.08% when the full dataset is used and only 63.12% and 63.23% accuracy for calcifications and masses respectively, contradicting the previous results. Because CNNs automatically learn features, it can be hard to know exactly what goes on underneath the hood of these models, especially when using architectures like VGG19 and MobileNetV2. Visualising heatmaps of the feature maps for each convolution layer could help understand why these models react differently when using all images or only specific types of mammograms, but is out of the scope of this project given the time frame.

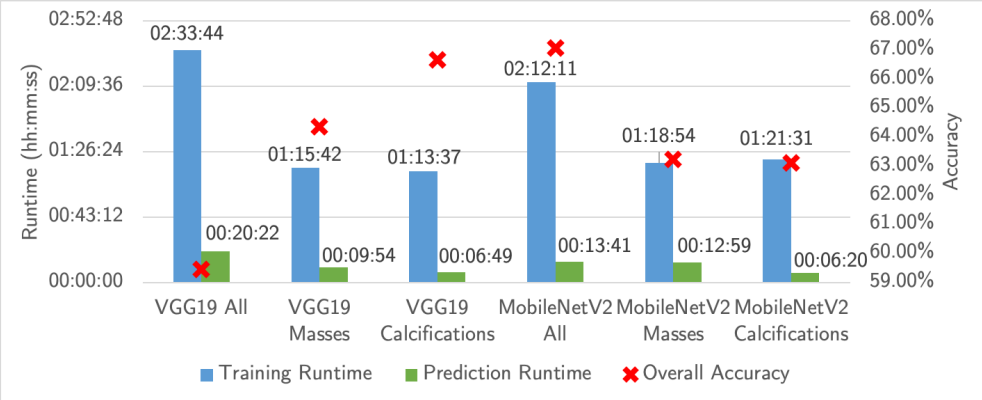


Figure 6.9: Training and prediction runtimes on the CBIS-DDSM dataset when using different mammogram types (all, masses and calcifications).

In terms of runtime, training and prediction times are approximately twice as fast, since there is roughly twice less data after the dataset split (see Figure 6.9).

6.9 Results Summary

The most interesting results from the bag-of-tricks approached are summarised in Figure 6.10, depicting each technique's accuracy relative to the benchmark.

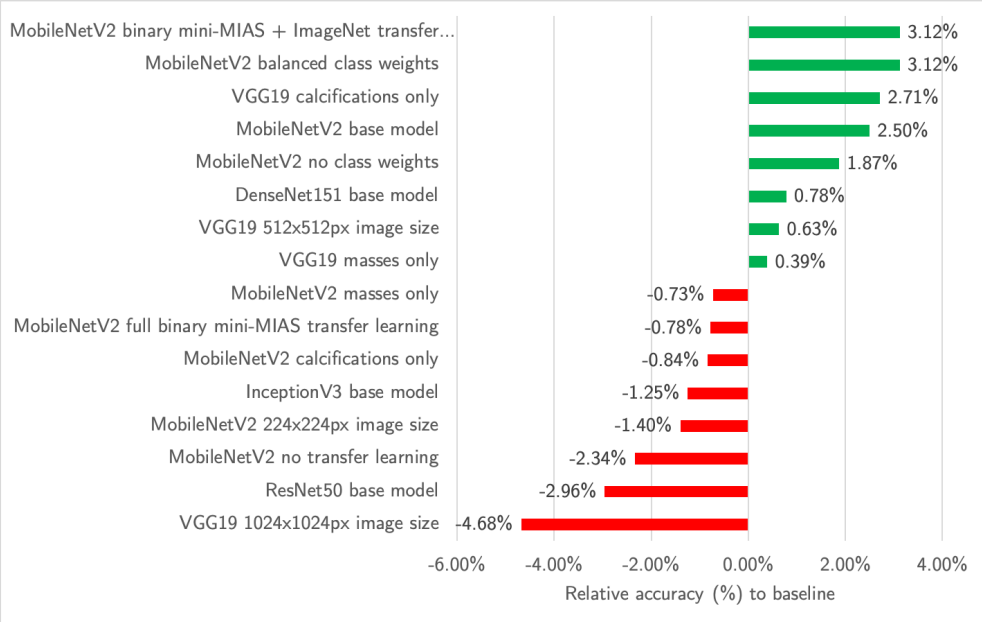


Figure 6.10: Bar chart summarising the relative accuracies achieved for each experiment compared to the baseline developed as group on the CBIS-DDSM dataset.

Chapter 7

Conclusions

7.1 Achievements

The main goal of this project was to design and implement a deep learning pipeline capable of detecting cases of breast cancer in mammograms through various deep learning techniques. After investigating a wide array of techniques using a bag-of-tricks approach, a fully-functional pipeline with data pre-processing steps, a CNN model for learning the data and prediction visualisations was created.

This deep learning pipeline exposed the effects of the different techniques used. The most positive result (67.08%) on CBIS-DDSM came from transfer learning techniques, using ImageNet weights with MobileNetV2 and binary mini-MIAS weights for the custom MLP layers, coupled with class weight techniques for balancing the dataset. Separating CBIS-DDSM samples between masses and calcifications also yielded increased accuracies compared to the benchmark (64.35% and 66.67% respectively) when using VGG19 as a base model. However, other techniques did not behave as expected and resulted in poor accuracies, such as using data augmentation on a small dataset (mini-MIAS), separating CBIS-DDSM samples between masses and calcifications with MobileNetV2 instead of VGG19 and using larger input images with extra convolutional and pooling layers to learn lower-level features.

7.2 Code Availability

Digital Object Identifiers (DOI) have been generated for the open-sourced code repositories to ensure that the code can be permanently identified and referenced on the web, as URLs can easily change over time while DOIs remain immutable.

The code developed for this dissertation can be found online at the following URL: `https://doi.org/10.5281/zenodo.3985051` (Jaamour et al., 2020*b*), while the code developed in common as a group at the beginning of the dissertation can be found online at the following URL: `https://doi.org/10.5281/zenodo.3975093` (Jaamour et al., 2020*a*).

7.3 Limitations

A known limitation concerning all breast cancer detection systems lies with the data. Indeed, the most widely used datasets of mammograms (e.g. DDSM) contain mammography data that mainly originates from white females located in North America (see Table 7.1), which naturally introduces bias to the model learning this data (Yala et al., 2019).

	Data source	
Race	**MGH**	**WFUSM**
Asian	2.06%	0.20%
Black	4.12%	20.40%
Spanish Surname	6.55%	1.80%
American Indian	0.00%	0.10%
Other	0.75%	0.10%
Unknown	30.34%	0.30%
White	**56.18%**	**77.00%**

Table 7.1: DDSM dataset patient population statistics (female). Data collected by Massachusetts General Hospital (MGH) and Wake Forest University School of Medicine (WFUSM) (Heath et al., 2001).

Different body types linked to the geographic location of the patients used to create these databases can have a direct impact on the mammograms themselves and not generalise to females from other cultures. For example, a recent study with 53,000 North American females showed how diets that include dairy milk consumption might increase the risk of breast cancer by a maximum of 80% based on the consumption (Fraser et al., 2020). This means that if these deep learning algorithms were implemented in clinics outside western countries, they might not generalise well to other body morphologies (e.g. due to different diets based on the geolocation's culture). This limitation could be resolved by collecting more varied data from multiple locations around the world, not just a single region, which would also help deep learning algorithms as more data is always welcomed.

Another limitation in terms of the detection system's usability is the confidence of the predictions. Indeed, when given new test samples, the model

predicts a class label, e.g. benign or malignant. However, these do not indicate the prediction's confidence, as it can be anywhere between the decision boundary's limit (not confident) and far from the decision boundary (confident). Therefore, from a clinical point of view, it is hard to make a decision based on the predictions made by a system similar to this one. Ideally, a probability-based confidence metric would be coupled with the predictions to motivate the next step after the diagnosis. For example, if the confidence of a malignant tumour is high (e.g. 99%), then breast-conserving surgery or chemotherapy can be recommended, whereas if the confidence is low (e.g. 54%), then further screening tests can be recommended instead.

Finally, the time frame of this project was a limiting factor in the final performance achieved as an extensive fine-tuning method like grid search would not have had the time to try different combinations of configurations and could not be implemented due to the issues mentioned in Section 4.2.2.

7.4 Future Work

The main area of work that requires improvements is the mammogram preprocessing as it is often an area where significant performance gains can be found (Litjens et al., 2017) by using techniques such as global contrast normalisation (GCN), local contrast normalisation, and Otsu's threshold segmentation. Artefacts such as tags on the x-rays and black backgrounds should all be removed using computer vision techniques to avoid having the CNN learn irrelevant features.

Another area where improvements can be made is the fine-tuning to extract better performance on the datasets and avoid overfitting. With the data-preprocessing mentioned above, images would be smaller (e.g. no redundant dark background), which would allow for quicker runtimes (the results in Section 6.6 revealed that smaller images lower the training runtime), which would allow fine-tuning algorithms like grid search to explore more combinations of configurations in order to unlock better solutions.

7.5 Reflections

This project was an exciting challenge from my point of view as it encompassed all the classical challenges that need to be faced when building deep learning algorithms, clearly showing that creating a solution with high performance is not as easy as it sounds. Having first-hand experienced a family member going through cancer, having the opportunity to use my knowledge to contribute to the field of cancer detection was motivating.

Bibliography

Abadi, M., Agarwal, A., Barham, P., Brevdo, E., Chen, Z., Citro, C., Corrado, G. S., Davis, A., Dean, J., Devin, M., Ghemawat, S., Goodfellow, I., Harp, A., Irving, G., Isard, M., Jia, Y., Jozefowicz, R., Kaiser, L., Kudlur, M., Levenberg, J., Mané, D., Monga, R., Moore, S., Murray, D., Olah, C., Schuster, M., Shlens, J., Steiner, B., Sutskever, I., Talwar, K., Tucker, P., Vanhoucke, V., Vasudevan, V., Viégas, F., Vinyals, O., Warden, P., Wattenberg, M., Wicke, M., Yu, Y. and Zheng, X. (2015), 'TensorFlow: Large-scale machine learning on heterogeneous systems'. Software available from tensorflow.org.
URL: *https://www.tensorflow.org/*

Akay, M. F. (2009), 'Support vector machines combined with feature selection for breast cancer diagnosis', *Expert Systems with Applications* **36**(2 PART 2), 3240–3247.
URL: *http://dx.doi.org/10.1016/j.eswa.2008.01.009*

Amari, S.-i. and Wu, S. (1999), 'Improving support vector machine classifiers by modifying kernel functions', *Neural Networks* **12**(6), 783–789.

American Cancer Society (2019), 'American Cancer Society screening recommendations for women at average breast cancer risk', `https://www.cancer.org/cancer/breast-cancer/screening-tests-and-early-detection/american-cancer-society-recommendations-for-the-early-detection-of-breast-cancer.html`. [Online] Accessed: 2020-06-22.

Asri, H., Mousannif, H., Al Moatassime, H. and Noel, T. (2016), Using Machine Learning Algorithms for Breast Cancer Risk Prediction and Diagnosis, *in* 'Procedia Computer Science', Vol. 83, Elsevier, pp. 1064–1069.

Bennett, K. P. and Blue, J. A. (1998), 'Support vector machine approach to decision trees', *IEEE International Conference on Neural Networks - Conference Proceedings* **3**, 2396–2401.

Bergstra, J., Yamins, D. and Cox, D. (2013), 'Hyperopt: A Python Library

for Optimizing the Hyperparameters of Machine Learning Algorithms', *Proceedings of the 12th Python in Science Conference* (Scipy), 13–19.

Bhardwaj, A. and Tiwari, A. (2015), 'Breast cancer diagnosis using Genetically Optimized Neural Network model', *Expert Systems with Applications* **42**(10), 4611–4620.

Cancer Research UK (2020), 'Breast cancer statistics', `https://www.cancerresearchuk.org/health-professional/cancer-statistics/statistics-by-cancer-type/breast-cancer`. [Online] Accessed: 2020-05-28.

Caruana, R. and Niculescu-Mizil, A. (2006), An empirical comparison of supervised learning algorithms, *in* 'ACM International Conference Proceeding Series', Vol. 148, ACM Press, New York, New York, USA, pp. 161–168.
URL: *http://portal.acm.org/citation.cfm?doid=1143844.1143865*

Caulfield, B. (2009), 'What's the Difference Between a CPU vs a GPU, NVIDIA Blog', `https://blogs.nvidia.com/blog/2009/12/16/whats-the-difference-between-a-cpu-and-a-gpu/`. [Online] Accessed: 2020-07-29.

Chen, Y., Zhang, Q., Wu, Y., Liu, B., Wang, M. and Lin, Y. (2019), Fine-tuning ResNet for breast cancer classification from mammography, *in* 'Lecture Notes in Electrical Engineering', Vol. 536, Springer Verlag, pp. 83–96.
URL: *https://doi.org/10.1007/978-981-13-6837-0{_}7*

Chollet, F. et al. (2015), 'Keras', `https://github.com/fchollet/keras`.

Clark, K., Vendt, B., Smith, K., Freymann, J., Kirby, J., Koppel, P., Moore, S., Phillips, S., Maffitt, D., Pringle, M., Tarbox, L. and Prior, F. (2013), 'The cancer imaging archive (TCIA): Maintaining and operating a public information repository', *Journal of Digital Imaging* **26**(6), 1045–1057.

Deng, J., Dong, W., Socher, R., Li, L.-J., Kai Li and Li Fei-Fei (2010), ImageNet: A large-scale hierarchical image database, *in* 'Institute of Electrical and Electronics Engineers (IEEE)', pp. 248–255.

Diaz, O., Marti, R., Llado, X. and Agarwal, R. (2018), Mass detection in mammograms using pre-trained deep learning models, *in* E. A. Krupinski, ed., '14th International Workshop on Breast Imaging (IWBI 2018)', Vol. 10718, SPIE, p. 12.
URL: *https://www.spiedigitallibrary.org/conference-proceedings-of-spie/10718/2317681/Mass-detection-in-mammograms-using-pre-trained-deep-learning-models/10.1117/12.2317681.full*

Dietterich, T. (1995), 'Overfitting and undercomputing in machine learning', *ACM computing surveys (CSUR)* **27**(3), 326–327.

Elter, M. and Horsch, A. (2009), 'CADx of mammographic masses and clustered microcalcifications: A review', *Medical Physics* **36**(6), 2052–2068.

Falconi, L. G., Perez, M. and Aguilar, W. G. (2019), 'Transfer Learning in Breast Mammogram Abnormalities Classification with Mobilenet and Nasnet', *International Conference on Systems, Signals, and Image Processing* **2019-June**, 109–114.

Fraser, G. E., Jaceldo-Siegl, K., Orlich, M., Mashchak, A., Sirirat, R. and Knutsen, S. (2020), 'Dairy, soy, and risk of breast cancer: those confounded milks', *International Journal of Epidemiology* .
URL: *https://academic.oup.com/ije/advance-article/doi/10.1093/ije/dyaa007/5743492*

Géron, A. (2019), *Hands-On Machine Learning with Scikit-Learn, Keras & TensorFlow*, 2nd edn, O'Reilly Media.

Hanlon, J. (2016), 'Why is so much memory needed for deep neural networks?', `https://www.graphcore.ai/posts/why-is-so-much-memory-needed-for-deep-neural-networks`. [Online] Accessed: 2020-07-30.

Heath, M., Bowyer, K., Kopans, D., Moore, R. and Kegelmeyer, W. P. (2001), The Digital Database for Screening Mammography, *in* 'Fifth International Workshop on Digital Mammography', Medical Physics Publishing, pp. 212–218.

Hepsağ, P. U., Özel, S. A. and Yazici, A. (2017), Using deep learning for mammography classification, *in* '2nd International Conference on Computer Science and Engineering, UBMK 2017', Institute of Electrical and Electronics Engineers Inc., pp. 418–423.

Hunter, J. D. (2007), 'Matplotlib: A 2d graphics environment', *Computing in Science & Engineering* **9**(3), 90–95.

Jaamour, A., Patel, A. and Chen, S.-J. (2020*a*), 'Breast cancer detection in mammograms using deep learning techniques - common pipeline code'.
URL: *https://doi.org/10.5281/zenodo.3975093*

Jaamour, A., Patel, A. and Chen, S.-J. (2020*b*), 'Breast cancer detection in mammograms using deep learning techniques - source code'.
URL: *https://doi.org/10.5281/zenodo.3985051*

Jadoon, M. M., Zhang, Q., Haq, I. U., Butt, S. and Jadoon, A. (2017), 'Three-Class Mammogram Classification Based on Descriptive CNN Features', *BioMed Research International* **2017**.

Karabatak, M. and Ince, M. C. (2009), 'An expert system for detection of breast cancer based on association rules and neural network', *Expert Systems with Applications* **36**(2 PART 2), 3465–3469.

Keras (2020), 'Keras Applications', `https://keras.io/api/applications/`. [Online] Accessed: 2020-08-09.

Kharya, S., Agrawal, S. and Soni, S. (2014), 'Naive Bayes Classifiers: A Probabilistic Detection Model for Breast Cancer', *International Journal of Computer Applications* **92**(10), 26–31.

Krizhevsky, A., Sutskever, I. and Hinton, G. E. (2012), ImageNet Classification with Deep Convolutional Neural Networks, Technical report, Google.
URL: *http://code.google.com/p/cuda-convnet/*

Kumar, U. K., Nikhil, M. B. and Sumangali, K. (2017), 'Prediction of breast cancer using voting classifier technique', *2017 IEEE International Conference on Smart Technologies and Management for Computing, Communication, Controls, Energy and Materials, ICSTM 2017 - Proceedings* (August), 108–114.

LeCun, Y., Bottou, L., Bengio, Y. and Haffner, P. (1998), 'Gradient-based learning applied to document recognition', *Proceedings of the IEEE* **86**(11), 2278–2323.

Lee, R. S., Gimenez, F., Hoogi, A., Miyake, K. K., Gorovoy, M. and Rubin, D. L. (2017), 'A curated mammography data set for use in computer-aided detection and diagnosis research', *Scientific Data* **4**.

Li, J. and Allinson, N. M. (2008), 'A comprehensive review of current local features for computer vision', *Neurocomputing* **71**(10-12), 1771–1787.

Litjens, G., Kooi, T., Bejnordi, B. E., Setio, A. A. A., Ciompi, F., Ghafoorian, M., van der Laak, J. A., van Ginneken, B. and Sánchez, C. I. (2017), 'A survey on deep learning in medical image analysis'.

Liu, X. Y., Wu, J. and Zhou, Z. H. (2009), 'Exploratory undersampling for class-imbalance learning', *IEEE Transactions on Systems, Man, and Cybernetics, Part B: Cybernetics* **39**(2), 539–550.

Martin, Laura J. (2019), 'WebMD: Breast Biopsy for Breast Cancer Diagnosis', `https://www.webmd.com/breast-cancer/breast-biopsy`. [Online] Accessed: 2020-06-22.

McCulloch, W. S. and Pitts, W. (1943), 'A logical calculus of the ideas immanent in nervous activity', *The bulletin of mathematical biophysics* **5**(4), 115–133.

Medjahed, S. A., Ait Saadi, T. and Benyettou, A. (2013), 'Breast Cancer Diagnosis by using k-Nearest Neighbor with Different Distances and Classification Rules', *International Journal of Computer Applications* **62**(1), 1–5.

Michael, W. (2020), 'seaborn: statistical data visualization', `https://seaborn.pydata.org/`. [Online] Accessed: 2020-08-09.

Montazeri, M., Montazeri, M., Montazeri, M. and Beigzadeh, A. (2016), 'Machine learning models in breast cancer survival prediction', *Technology and Health Care* **24**(1), 31–42.

Oliphant, T. and developers, N. (2020), 'NumPy', `https://numpy.org/`. [Online] Accessed: 2020-08-09.

Osareh, A. and Shadgar, B. (2010), Machine learning techniques to diagnose breast cancer, *in* '2010 5th International Symposium on Health Informatics and Bioinformatics, HIBIT 2010', pp. 114–120.

Paliwal, M. and Kumar, U. A. (2009), 'Neural networks and statistical techniques: A review of applications'.

pandas development team, T. (2020), 'pandas-dev/pandas: Pandas'.
URL: *https://doi.org/10.5281/zenodo.3509134*

Paszke, A., Gross, S., Massa, F., Lerer, A., Bradbury, J., Chanan, G., Killeen, T., Lin, Z., Gimelshein, N., Antiga, L., Desmaison, A., Kopf, A., Yang, E., DeVito, Z., Raison, M., Tejani, A., Chilamkurthy, S., Steiner, B., Fang, L., Bai, J. and Chintala, S. (2019), Pytorch: An imperative style, high-performance deep learning library, *in* H. Wallach, H. Larochelle, A. Beygelzimer, F. d'Alché-Buc, E. Fox and R. Garnett, eds, 'Advances in Neural Information Processing Systems 32', Curran Associates, Inc., pp. 8024–8035.
URL: *http://papers.neurips.cc/paper/9015-pytorch-an-imperative-style-high-performance-deep-learning-library.pdf*

Pedregosa, F., Varoquaux, G., Gramfort, A., Michel, V., Thirion, B., Grisel, O., Blondel, M., Prettenhofer, P., Weiss, R., Dubourg, V., Vanderplas, J., Passos, A., Cournapeau, D., Brucher, M., Perrot, M. and Duchesnay, E. (2011), 'Scikit-learn: Machine learning in Python', *Journal of Machine Learning Research* **12**, 2825–2830.

Peterson, L. E. (2009), 'K-nearest neighbor', *Scholarpedia* **4**(2), 1883.

Polat, K. and Güneş, S. (2007), 'Breast cancer diagnosis using least square support vector machine', *Digital Signal Processing* **17**(4), 694–701.
URL: *https://linkinghub.elsevier.com/retrieve/pii/S1051200406001461*

Punitha, S., Amuthan, A. and Joseph, K. S. (2018), 'Benign and malignant breast cancer segmentation using optimized region growing technique', *Future Computing and Informatics Journal* **3**(2), 348–358.

Quinlan, J. R. (2014), *C4. 5: programs for machine learning*, Elsevier.

Ramos-Pollán, R., Guevara-López, M. A., Suárez-Ortega, C., Díaz-Herrero, G., Franco-Valiente, J. M., Rubio-Del-Solar, M., González-De-Posada, N., Vaz, M. A. P., Loureiro, J. and Ramos, I. (2012), 'Discovering mammography-based machine learning classifiers for breast cancer diagnosis', *Journal of Medical Systems* **36**(4), 2259–2269.

Raschka, S. and Mirjalili, V. (2017), *Python machine learning*, Packt Publishing Ltd.

Rish, I. et al. (2001), An empirical study of the naive bayes classifier, *in* 'IJCAI 2001 workshop on empirical methods in artificial intelligence', Vol. 3, pp. 41–46.

Russell, S. and Norvig, P. (2002), *Artificial intelligence: a modern approach*, Pearson.

Sarkar, M. and Leong, T. Y. (2000), 'Application of K-nearest neighbors algorithm on breast cancer diagnosis problem.', *Proceedings / AMIA ... Annual Symposium. AMIA Symposium* pp. 759–763.

Sarosa, S. J. A., Utaminingrum, F. and Bachtiar, F. A. (2018), Mammogram Breast Cancer Classification Using Gray-Level Co-Occurrence Matrix and Support Vector Machine, *in* '3rd International Conference on Sustainable Information Engineering and Technology, SIET 2018 - Proceedings', Institute of Electrical and Electronics Engineers Inc., pp. 54–59.

Scikit-Learn Documentation: Neural network models (supervised) (2019), `https://scikit-learn.org/stable/modules/neural_networks_supervised.html`. [Online] Accessed: 2020-06-22.

Shen, L., Margolies, L. R., Rothstein, J. H., Fluder, E., McBride, R. B. and Sieh, W. (2017), 'Deep Learning to Improve Breast Cancer Early Detection on Screening Mammography', *Scientific Reports* **9**(1).
URL: *http://arxiv.org/abs/1708.09427http://dx.doi.org/10.1038/s41598-019-48995-4*

Simonyan, K. and Zisserman, A. (2014), 'Very deep convolutional networks for large-scale image recognition', *arXiv preprint arXiv:1409.1556* .

Srivastava, N., Hinton, G., Krizhevsky, A. and Salakhutdinov, R. (2014), Dropout: A Simple Way to Prevent Neural Networks from Overfitting, Technical report.

Suckling, J. (1994), 'The mammographic image analysis society digital mammogram database exerpta medica', *International Congress Series* **1069**, 375–378.
URL: *http://peipa.essex.ac.uk/info/mias.html*

Sumbaly, R., Vishnusri, N. and Jeyalatha, S. (2014), 'Diagnosis of Breast Cancer using Decision Tree Data Mining Technique', *International Journal of Computer Applications* **98**(10), 16–24.

Szeliski, R. (2010), *Computer vision: algorithms and applications*, Springer Science & Business Media.

Vishrutha, V. and Ravishankar, M. (2014), Early detection and classification of breast cancer, *in* 'Advances in Intelligent Systems and Computing', Vol. 327, Springer Verlag, pp. 413–419.
URL: *https://link.springer.com/chapter/10.1007/978-3-319-11933-5{_}45*

Wang, D., Khosla, A., Gargeya, R., Irshad, H. and Beck, A. H. (2016), 'Deep Learning for Identifying Metastatic Breast Cancer'.
URL: *http://arxiv.org/abs/1606.05718*

What Mammograms Show: Calcifications, Cysts, Fibroadenomas (2018), `https://www.breastcancer.org/symptoms/testing/types/mammograms/mamm_show`. [Online] Accessed: 2020-06-21.

Wolberg, W. H., Street, W. N. and Mangasarian, O. L. (1995), 'UCI Machine Learning Repository: Breast Cancer Wisconsin (Diagnostic) Data Set', `https://archive.ics.uci.edu/ml/datasets/Breast+Cancer+Wisconsin+{%}28Diagnostic{%}29`. [Online] Accessed: 2020-06-16.

Wu, Y., Giger, M. L., Doi, K., Vyborny, C. J., Schmidt, R. A. and Metz, C. E. (1993), 'Artificial neural networks in mammography: Application to decision making in the diagnosis of breast cancer', *Radiology* **187**(1), 81–87.
URL: *https://pubs.rsna.org/doi/abs/10.1148/radiology.187.1.8451441*

Yala, A., Lehman, C., Schuster, T., Portnoi, T. and Barzilay, R. (2019), 'A Deep Learning Mammography-based Model for Improved Breast Cancer

Risk Prediction', *Radiology* **292**(1), 60–66.
URL: *http://pubs.rsna.org/doi/10.1148/radiol.2019182716*

Yue, W., Wang, Z., Chen, H., Payne, A. and Liu, X. (2018), 'Machine Learning with Applications in Breast Cancer Diagnosis and Prognosis', *Designs* **2**(2), 13.
URL: *http://www.mdpi.com/2411-9660/2/2/13*

Zhu, M., Xia, J., Jin, X., Yan, M., Cai, G., Yan, J. and Ning, G. (2018), 'Class weights random forest algorithm for processing class imbalanced medical data', *IEEE Access* **6**, 4641–4652.

Appendix A

Ethical Application Approval Letter

Approval letter received on 18/06/2020 from the University of St Andrew's Teaching and Research Ethics Committee concerning the ethical application submitted on 03/06/2020 for this research project.

University Teaching and Research Ethics Committee

18 June 2020

Dear Shuen-Jen, Adam and Ashay,

Thank you for submitting your ethical application, which was considered by the School of Computer Science Ethics Committee, where the following documents were reviewed:

1. Ethical Application Form

The School of Computer Science Ethics Committee has been delegated to act on behalf of the University Teaching and Research Ethics Committee (UTREC) and has granted this application ethical approval. The particulars relating to the approved project are as follows -

Approval Code:	CS14950	**Approved on:**	18.06.20	**Approval Expiry:**	18.06.25
Project Title:	Breast Cancer Detection in Mammograms using Deep Learning Techniques				
Researcher(s):	Shuen-Jen Chen, Adam Jaamour and Ashay Patel				
Supervisor(s):	Dr David Harris-Birtill				

Approval is awarded for five years. Projects which have not commenced within two years of approval must be re-submitted for review by your School Ethics Committee. If you are unable to complete your research within the five year approval period, you are required to write to your School Ethics Committee Convener to request a discretionary extension of no greater than 6 months or to re-apply if directed to do so, and you should inform your School Ethics Committee when your project reaches completion.

If you make any changes to the project outlined in your approved ethical application form, you should inform your supervisor and seek advice on the ethical implications of those changes from the School Ethics Convener who may advise you to complete and submit an ethical amendment form for review.

Any adverse incident which occurs during the course of conducting your research must be reported immediately to the School Ethics Committee who will advise you on the appropriate action to be taken.

Approval is given on the understanding that you conduct your research as outlined in your application and in compliance with UTREC Guidelines and Policies (http://www.st-andrews.ac.uk/utrec/guidelinespolicies/). You are also advised to ensure that you procure and handle your research data within the provisions of the Data Provision Act 1998 and in accordance with any conditions of funding incumbent upon you.

Yours sincerely

Wendy Boyter

School Ethics Committee Administrator

Appendix B

Languages & Frameworks Comparison

B.1 Programming Languages

In terms of speed, compiled languages are quicker than interpreted languages, but because the main bottleneck in a deep learning system is the training phase of the model, which mainly relies on the library used rather than the language itself, speed is not taken into account.

	Python	**Java**	**R**	**Javascript**
Familiarity	Extremely familiar	Familiar	Unfamiliar	Familiar
Deep Learning Libraries support	Tensorflow, Keras, PyTorch + ML suite (NumPy, Pandas, MatplotLib, Pandas)	DL4J (Deeplearning4j)	Tensorflow, Keras	Tensorflow
Third-party CUDA support	Yes	Yes	Yes	Yes
Pre-trained models support	Yes	Yes	Yes	Yes
CNN support	Yes	Yes	Yes	Yes
Speed	Slow (interpreted)	Fast (compiled)	Fast (compiled)	Slow (interpreted)

Table B.1: Table comparing the pros and cons of different programming languages when implementing a deep learning system.

B.2 Deep Learning Frameworks

Figures source: `https://keras.io/why_keras/`.

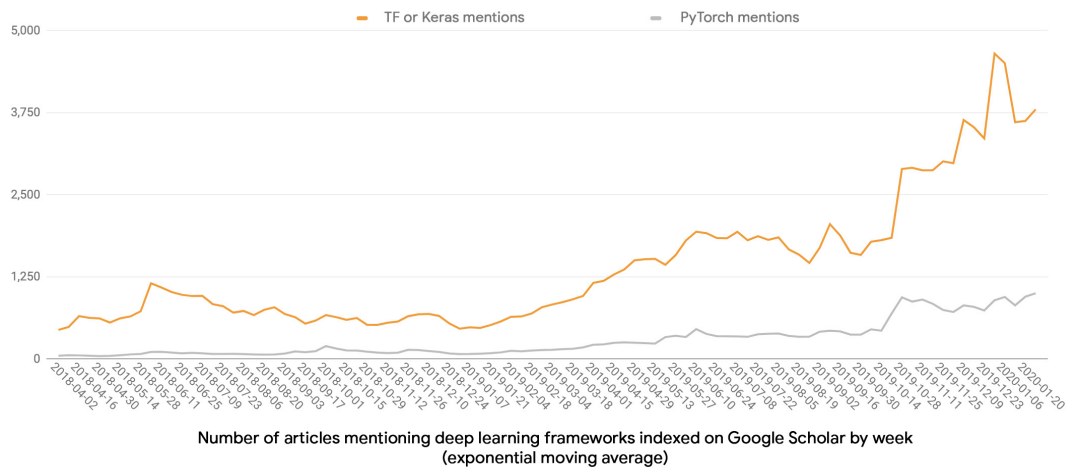


Figure B.1: Chart highlighting the number of articles mentioning Keras and PyTorch. Figure downloaded from Keras website.

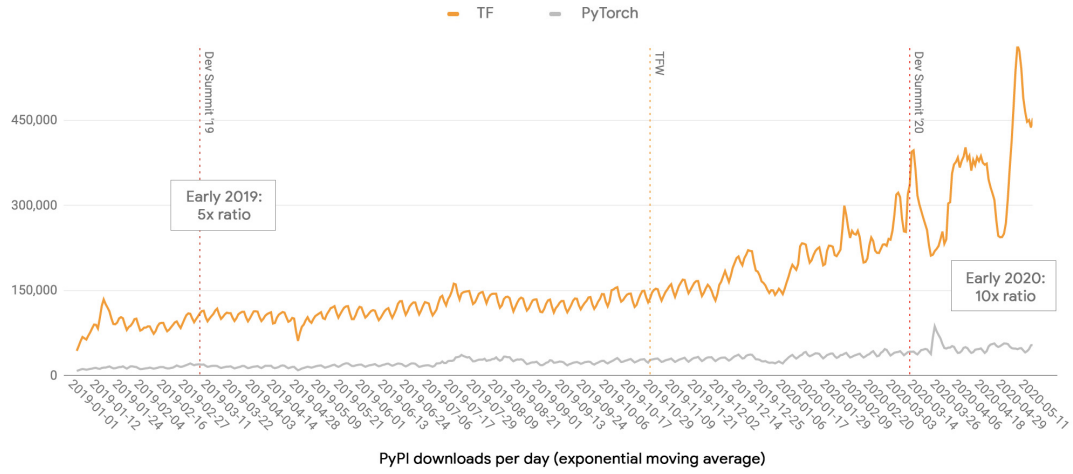


Figure B.2: Chart depicting the number of PyPI downloads for Keras and PyTorch. Figure downloaded from Keras website.

Appendix C

Usage Instructions

C.1 Installation Instructions

Start by cloning the GitHub repository (either the common or the individual code):

```
cd ~/Projects
git clone https://github.com/Adamouization/Breast-Cancer-Detection
    -Code
```

Create a repository that will be used to install Tensorflow 2 with CUDA 10 for Python and activate the virtual environment for GPU usage:

```
cd libraries/tf2
tar xvzf tensorflow2-cuda-10-1-e5bd53b3b5e6.tar.gz
sh build.sh
```

Activate the virtual environment:

```
source /Breast-Cancer-Detection-Code/tf2/venv/bin/activate
```

"*cd*" into the "*src*" directory and run the code below.

C.2 Individual Code Instructions

Run the code:

```
main.py [-h] -d DATASET [-mt MAMMOGRAMTYPE] -m MODEL [-r RUNMODE]
    [-lr LEARNING_RATE] [-b BATCHSIZE] [-e1 MAX_EPOCH_FROZEN] [-e2
    MAX_EPOCH_UNFROZEN] [-gs] [-roi] [-v]
```

where:

- *-h* is a flag for help on how to run the code.
- *DATASET* is the dataset to use. Must be either *mini-MIAS*, *mini-MIAS-binary* or *CBIS-DDMS*.

- *MAMMOGRAMTYPE* is the type of mammogram to use. Can be either *calc*, *mass* or *all*.
- *MODEL* is the model to use. Must be either *VGG*, *VGG-common*, *Inception* or *CNN*. Default value is *VGG-common*.
- *RUNMODE* is the learning rate used for the all the non-pre-trained ImageNet layers. Defaults to 1e-3. Must be a float.
- *LEARNING_RATE* is the optimiser's initial learning rate to use when training the model during the first phase (frozen layers). Defaults to *0.001*. Must be a positive float.
- *BATCHSIZE* is the batch size to use when training the model. Defaults to *2*. Must be a positive integer.
- *MAXEPOCHFROZEN* is the maximum number of epochs in the first training phrase (with frozen layers). Defaults to *100*. Must be a positive integer.
- *MAXEPOCHUNFROZEN* is the maximum number of epochs in the second training phrase (with unfrozen layers). Defaults to *50*. Must be a positive integer.
- *-roi* is a flag to use only cropped versions of the images around the ROI. Only usable with mini-MIAS dataset. Defaults to *False*.
- *-v* is a flag controlling verbose mode, which prints additional statements for debugging purposes. Defaults to *False*.

C.3 Common Pipeline Code Instructions

Run the code:

```
python main.py [-h] -d DATASET -m MODEL [-r RUNMODE] [-i IMAGESIZE
    ] [-v]
```

where:

- *-h* is a flag for help on how to run the code.
- *DATASET* is the dataset to use. Must be either *mini-MIAS* or *CBIS-DDMS*.
- *MODEL* is the model to use. Must be either *basic* or *advanced*.
- *RUNMODE* is the mode to run in (*train* or *test*). Default value is *train*.
- *IMAGESIZE* is the image size to feed into the CNN model (*small* - 512x512px; or *large* - 2048x2048px). Default value is *small*.

- *-v* is a flag controlling verbose mode, which prints additional statements for debugging purposes.

C.4 Dataset Installation Instructions

C.4.1 mini-MIAS dataset

This example will use the mini-MIAS dataset[1]. After cloning the project, travel to the *data/mini-MIAS* directory (there should be 3 files in it).

Create *images_original* and *images_processed* directories in this directory:

```
cd data/mini-MIAS/
mkdir images_original
mkdir images_processed
```

Move to the *images_original* directory and download the raw un-processed images:

```
cd images_original
wget http://peipa.essex.ac.uk/pix/mias/all-mias.tar.gz
```

Unzip the dataset then delete all non-image files:

```
tar xvzf all-mias.tar.gz
rm -rf *.txt
rm -rf README
```

Move back up one level and move to the *images_processed* directory. Create 3 new directories there (*benign_cases*, *malignant_cases* and *normal_cases*):

```
cd ../images_processed
mkdir benign_cases
mkdir malignant_cases
mkdir normal_cases
```

Now run the python script for processing the dataset and render it usable with Tensorflow and Keras:

```
python3 ../../../src/data_manipulations/mini-MIAS-initial-pre-
    processing.py
```

C.4.2 CBIS-DDSM dataset

These datasets are very large (exceeding 160GB) and more complex than the mini-MIAS dataset to use. They were downloaded by the University of St Andrews School of Computer Science computing officers onto *BigTMP*, a 15TB filesystem that is mounted on the Centos 7 computer lab clients with NVIDIA

[1] mini-MIAS dataset: `http://peipa.essex.ac.uk/info/mias.html`

GPUs usually used for storing large working data sets. Therefore, the download process of these datasets will not be covered in these instructions.

The generated CSV files to use these datasets can be found in the */data/CBIS-DDSM* directory, but the mammograms will have to be downloaded separately directly from the source. The CBIS-DDSM dataset can be downloaded here: `https://wiki.cancerimagingarchive.net/display/Public/CBIS-DDSM#5e40bd1f79d64f04b40cac57ceca9272`.

Appendix D

Remote Work Environment

Due to the coronavirus pandemic that started in March 2020, and has been ongoing since the beginning of this research project (June 2020 to August 2020), work had to be conducted remotely.

D.1 Coding environment

Due to the physical lab machines equipped with the GPUs being inaccessible during the pandemic, these machines had to be remotely accessed through SSH. As it is not efficient to implement an entire deep learning pipeline through a command-line interface, a *Jupyter Lab* session was created on a lab computer port and forwarded back to a local personal computer.

This is done by first logging onto the lab machine via SSH, activating the virtual environment and launching the Jupyter Lab session (these instructions are added into a bash script to be quickly executed):

```
ssh agj6@agj6.host.cs.st-andrews.ac.uk -t ssh agj6@pc5-026-l.cs.st
    -andrews.ac.uk
source /cs/scratch/agj6/tf2/venv/bin/activate
cd ~/Projects/Breast-Cancer-Detection-and-Segmentation
jupyter lab --no-browser --port=8888
```

Keeping the first SSH session alive, a second terminal session is opened, and the port forwarding from the lab machine to the personal computer is carried out using the following command:

```
nohup ssh -J agj6@agj6.host.cs.st-andrews.ac.uk agj6@pc5-026-l.cs.
    st-andrews.ac.uk -L 8888:localhost:8888 -N
```

To prevent the two SSH sessions from being automatically killed, some SSH settings are added to the "*/etc/ssh/ssh_config*" file, making the personal computer send a null packet to the lab machine every 5 minutes:

```
Host *
```

```
    ServerAliveInterval 300
    ServerAliveCountMax 2
```

Finally, `http://localhost:8888` is visited on a web browser to gain access to the Jupyter Lab interface, containing a menu bar, a file explorer, code editor and command line for running python files (see Figure D.1).

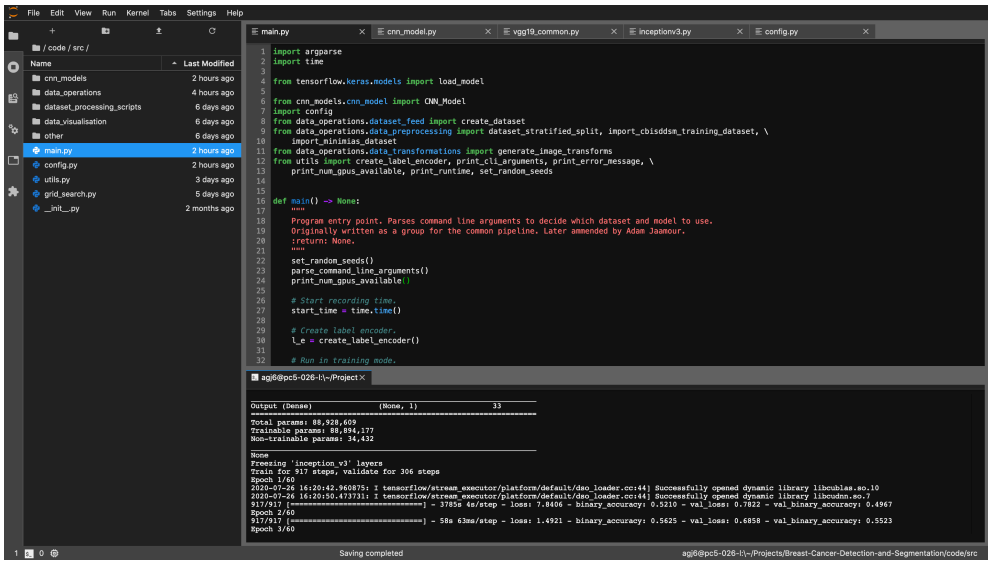


Figure D.1: Screenshot of the Jupyter Lab interface used to implement the project.

D.2 Code collaboration

For the group work part of the project, the code was collaboratively written and version controlled using GitHub, making use of the pull requests and merging features to combine code written by each group member. The code developed as a group can be found online on GitHub: `https://github.com/Adamouization/Breast-Cancer-Detection-Code`, while the code developed individually can be found online on GitHub as well: `https://github.com/Adamouization/Breast-Cancer-Detection-Mammogram-Deep-Learning`.

D.3 Supervisor meetings

Weekly video meetings with the project supervisor, Dr David Harris-Birtill, and the project co-supervisor, Lewis McMillan, were planned once per week via Microsoft Teams. Additional meetings with the other group members, Ashay Patel and Shuen-Jen Chen, were conducted via Microsoft Teams as well.

Appendix E

Team Meeting Summaries

This appendix contains the summary of the team meetings carried out during the development of the common deep learning pipeline from 24/06/2020 to 09/07/2020. They include the attendance, the matters discussed and the tasks set for the next meeting,

CS5098 MSc Dissertations
MS Teams Meetings

All meetings were conducted via MS Teams during the development of the common deep learning pipeline (from 24/06 to 09/07).

Team Members

- Adam Jaamour
- Ashay Patel
- Shuen-Jen Shen

Legend

Colour	Task	Attendance
	Task completed	Attended meeting
	Task incomplete	Missed meeting

24/06/2020 - MS Teams Meeting #1

Attendance & Task Completion

Name	Present	Task completed?
Adam	Yes	N/A
Ashay	Yes	N/A
Shuen-Jen	Yes	N/A

Issues discussed & Tasks carried out

- Set out tasks on what to implement for the next meeting for a very basic DL pipeline with data pre-processing, a pre-trained CNN model and some output visualisation.
- Setup common GitHub repo and working environment in separate branches.
- Queried St Andrews FixIt service for downloading CBIS-DDMS dataset on lab machine.

Tasks set for 29/06

Name	Tasks
Adam	Data pre-processing using mini-MIAS dataset for now: ● Initial pre-processing (CSV, script to organise images into class folders) ● Create usable dataset (images mapped to labels) ● Resize images for VGG19 CNN model input and split datasets ● (investigate data augmentation) ● (investigate how to remove artefacts)
Ashay	Implement VGG19 through Keras (on any data for now). Try 2 models: ● VGG19 on its own (using downsampled images) ● VGG19 with extra convolution/pooling layers using 512px image
Shuen-Jen	Take output from any template neural network from TF/Keras and visualise the output metrics using matplotlib and seaborn Output metrics to create: ROC, AUC, Overall accuracy to compare with other papers, Confusion matrix for classification
All	Download CBIS-DDSM dataset (send email to FixIt to either download after connecting to VPN, or to copy from Shuen-jen's lab computer).

29/06/2020 - MS Teams Meeting #2 (code review 1)

Attendance & Task Completion

Name	Present	Task completed?
Adam	Yes	Yes
Ashay	Yes	Yes
Shuen-Jen	Yes	Yes

Issues discussed & Tasks carried out

- Merged all branches to master
- Combined the output of data pre-processing (Adam) to the input for the VGG19 model (Ashay)
- Combined the output of the CNN model (Ashay) to the input of the result visualisation function (Shuen-jen)
- Applied mini-MIAS dataset transformation for Ashay and Shuen-Jen and tested/debugged full pipeline on GPU, refactored and reorganised some of the code.

Tasks set for 01/07

Name	Tasks
Adam	Reach out to IT about CBIS-DDSM dataset. Code improvements (normalise input images, include model name in output figure filename). Update README instructions for running the code.
Ashay	Run training over more epoch for wednesday meeting
Shuen-Jen	Formatting the Confusion Matrix
All	Research GPU memory usage for multiple training cycles

30/06/2020 - MS Teams Meeting #3 (code review 2)

Attendance & Task Completion

Name	Present	Task completed?
Adam	Yes	Yes
Ashay	Yes	Yes
Shuen-Jen	Yes	Yes

Issues discussed & Tasks carried out

- Stuart replied that the DDSM dataset is available on BigTMP and CBIS-DDSM is still downloading and will be uploaded right after.
- Merged code and ensured it worked fine
- Checked that the DDSM dataset was accessible from BigTMP

Tasks set for 01/07

Name	Tasks
Adam	Research gpu memory usage for multiple training cycles Investigate github commits author issue
Ashay	Transforms code & shuffle data when splitting
Shuen-Jen	Rotate labels (e.g. 45 degrees)
All	N/A

01/07/2020 - MS Teams Meeting #4

Attendance & Task Completion

Name	Present	Task completed?
Adam	Yes	Yes
Ashay	Yes	Yes
Shuen-Jen	Yes	Yes

Issues discussed & Tasks carried out

- CBIS-DDSM dataset successfully downloaded on BigTMP.
- Carry out changes recommended by David:
    - Do not use test dataset
    - Use same train/test/validation splits as other papers
    - Use validation output in confusion matrices
- Presentation:
    - Created presentation slides
    - Rehearsed presentation

No task set

02/07/2020 - MS Teams Meeting #5

Attendance & Task Completion

Name	Present	Task completed?
Adam	Yes	N/A
Ashay	Yes	N/A
Shuen-Jen	Yes	N/A

Issues discussed & Tasks carried out

- Gave our presentation for team DHB.
- Discussed tasks for next week: need to process large CBIS-DDSM dataset rather than mini-MIAS. Model and outputs are good enough for now and don't need to be touched.
- Both CBIS-DDSM and DDSM datasets have been downloaded and available at */cs/tmp/datasets/CBIS-DDSM*.

Tasks set for 07/07

Name	Tasks
Adam	Create dummy data using the mini-MIAS data set (similar CSV to the one Shuen-Jen is creating) Research data generators in Keras and batch data processing
Ashay	
Shuen-Jen	Generate CSV file with all paths that we need to use: ● 1st column is path to the image and the 2nd other column is the label ● Use the CSV files that are already split in training/testing sets ● (need to append the */cs/tmp/datasets/CBIS-DDSM* to the path) ● https://wiki.cancerimagingarchive.net/display/Public/CBIS-DDSM#5e40bd1f79d64f04b40cac57ceca9272
All	N/A

07/07/2020 - MS Teams Meeting #6 (code review 3)

Attendance & Task Completion

Name	Present	Task completed?
Adam	Yes	Yes
Ashay	Yes	Yes
Shuen-Jen	Yes	Yes

Issues discussed & Tasks carried out

- Merged code running on CBIS-DDSM dataset.
- Fix predictions for binary tasks (probability instead of class) and test the whole pipeline on both CBIS-DDSM and mini-MIAS datasets.
- Problems to consider for future code developments:
    - Maintain aspect ratio when resizing
    - Figure output saving, differentiate between binary and multi
    - Monitor accuracy/loss changes for CBIS-DDSM
    - Investigate out of memory warnings (doesn't appear with small batch size = 5 on mini-MIAS)

Tasks set for 08/07

Name	Tasks
Adam	Run basic model on CBIS-DDSM
Ashay	Run advanced model on CBIS-DDSM
Shuen-Jen	Run basic model on CBIS-DDSM
All	Saved outputs to google drive

08/07/2020 - MS Teams Meeting #7

Attendance & Task Completion

Name	Present	Task completed?
Adam	Yes	Yes
Ashay	Yes	Yes
Shuen-Jen	Yes	Yes

Issues discussed & Tasks carried out

- Meeting with David & Lewis
- Can ignore TF warnings
- Improvements to consider (based on David's suggestions):
    - Future: combine images from the same cases into 2 channels?
    - Instead of resizing image to VGG size, use conv layers at the beginning to resize (still need to make it square). Resize to 2048x2048px with padding to maintain aspect ratio, use 2 conv layers with stride 2 to get to 512x512px.
    - Complete the list of papers we are comparing with, including section of image used (ROI or whole image), type of CNN, dataset used and what type of image used (everything or just calc/mass/...), image size (original or resized), training/test/validation split.

No tasks set

09/07/2020 - MS Teams Meeting #8

Attendance & Task Completion

Name	Present	Task completed?
Adam	Yes	N/A
Ashay	Yes	N/A
Shuen-Jen	Yes	N/A

Issues discussed & Tasks carried out

- Merged code refactoring (adam) and large image sizes (ashay) and tested the results.
- Refactored output file names for better organisation.
- Use epoch size 100/50
- Upload all results so far to the shared “results” directory on our shared Google Drive.

Tasks set for 10/07

Name	Tasks
Adam	Run CBIS-DDSM basic model with large images (batch size 2)
Ashay	Run CBIS-DDSM advanced model with large images (batch size 2)
Shuen-Jen	Run CBIS-DDSM basic model with large images (batch size 1) Add 2 papers to the excel sheet of papers
All	Setup private repositories to individual development

Appendix F

Coding Project Structure

Structure of the individual implementation found in Figure F.1.

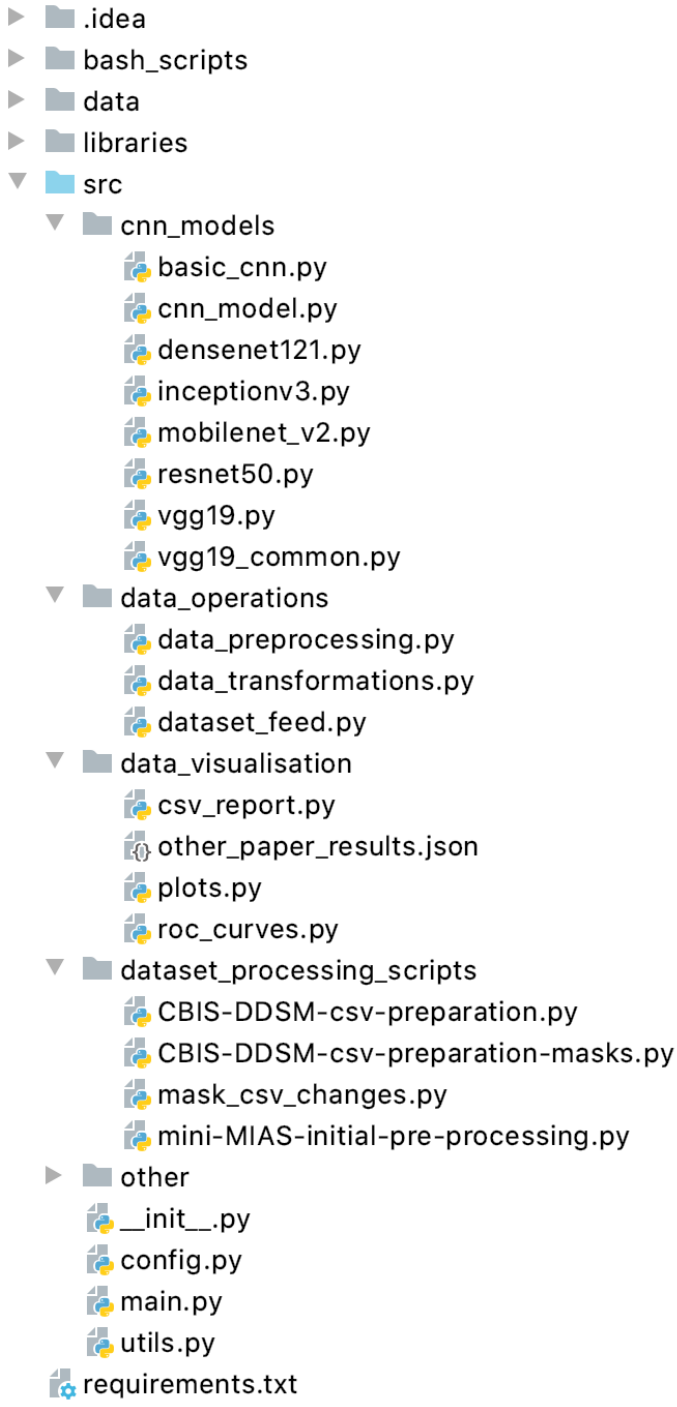


Figure F.1: Screenshot of the project structure.