Deep Learning Breast Cancer Detection

Final Year Project

Isaac Cilia Attard

A project report presented for the degree of
Bachelor of Science in Computer Science

Department of Computing
University of London
UK
July 18, 2025

Deep Learning Breast Cancer Detection

Final Year Project

Isaac Cilia Attard

Abstract

This project explores the application of convolutional neural networks (CNNs), and other architectures for detecting breast cancer using the Digital Database for Screening Mammography (DDSM) dataset. The devised system will incorporate data preprocessing, model training, and explainability through Grad-CAM heatmaps, with a Django-powered web application for inference. The study evaluates the model's feasibility for clinical integration, indicating the potential difficulties in adopting such systems like class imbalance, and computational constraints. Future work will involve model optimisation, more advanced architecures, and enhanced data augmentation to aid in generalisability, and usability.

Contents

1. Introduction

Breast cancer remains as one of the leading causes of cancer mortality amongst women around the globe. Routine screening processes can prompt early detection of cancer, and significantly improve patient outcomes, yet they are time-consuming, and success rates vary depending on the observer. Across the world, many hospitals are taking part in large-scale research programmes to investigate the feasibility of using deep learning systems, like convolutional neural networks (CNNs), to match or exceed the diagnostic acumen of human radiologists in breast cancer screening.

Upon the premise of this initiative, project 3.2 titled —Deep Learning Breast Cancer Detection, explores the potential of applying CNN-based models to publicly available datasets such as the Digital Database for Screening Mammography (DDSM), as well as other technologies like vision transformers, and transfer learning. The objective of this project is two-fold: to develop a working proof-of-concept that achieves statistical significance in its results on the DDSM dataset, and to evaluate the integration of such a prototype into existing clinical workflows, identifying potential challenges and limitations.

This introduction begins by stating the clinical, and technical motivation for this project, then presents research questions, and objectives, as well as an outline of the report structure.

1.1. Motivation

Mammography is widely known as the de facto standard for early breast cancer screening. Despite this, the visual similarity between benign, and malignant lesions can challenge even the most experienced of radiologists, leading to skewed results, and misdiagnosis. Often, false positives lead to unnecessary follow-ups, and false negatives result in delayed treatment, which can prove deadly. Deep learning can significantly reduce these issues by learning from large collections of labelled scans, complementing radiologists, and improving the accuracy of mammography.

Convolutional neural networks (CNNs), can be used to automatically extract features from an image, including edges, and complex textures, that can prove demonstrably effective at weeding out subtle patterns in pathology. Various studies have demonstrated the prowess of CNNs in tasks like tumour segmentation, and classification on medical imaging modalities like CT, and MRI scans, however, applications to mammography remain relatively untested, in part due to the challenges like standardisation, and generalisation.

The DDSM dataset is a large-scale collection of digital mammograms, annotated by radiologists for the presence or absence of malignant lesions. It is widely used in research, allowing for the effective benchmarking on model performance against established studies, and comparing CNN outputs to ground-truth labels. In addition, the use of the DDSM ensures that the prototype is developed on a dataset that is

representative of real-world clinical data, and is reproducible, allowing for further extensions, and collaborations.

Aside from technical performance, an automated system must be understood in how it could augment clinical workflows. Thus, a CNN that is reliable enough, could be used as an assistant to radiologists, prioritising high-risk cases for human review, and reducing the burden of radiologist workload, and turnaround times. On the other hand, understanding how the system oculd fail, like misclassification issues, could guide refinements in data preprocessing pipelines, model architecture, and evaluation metrics.

1.2. Research Questions, and Objectives

This project is comprised of the following, central research questions.

1. **Model Feasibility** — Can a CNN trained on the DDSM dataset achieve classification metrics (like AUC, sensitivity, specificity) that approach or exceed those for human readers under similar conditions?
2. **Prototype Evaluation** — What do preliminary results demonstrate about the feasibility of integrating a CNN-based system into clinical workflows? How might these insights inform augmentations to model architecture or data augmentation methods to improve the system?
3. **Clinical Integration** — What are some of the practical considerations, like inference time, interpretability, and failures that would affect the adoption of such deep learning systems in a clinical setting?

In a bid to answer these questions, the following will be performed.

- Develop a CNN architecture using TensorFlow, and Keras, optimised for binary classification of patches, using practices like dataset splitting, transfer learning, and regularisation.
- Train the model on the DDSM dataset, and report performance on a test partition using metrics like accuracy, sensitivity, specificity, and AUC to determine diagnostic accuracy.
- Analyse the model's performance to identify the categories of issues where performance degrades, and identify mitigation strategies.
- Reflect on integration into clinical settings, discussing how such a system helps to reduce radiologist workload in reading mammograms, and error rates.

1.3. Report Structure

- **Chapter 1** — Introduction — This chapter introduces the motivation for the project, and outlines the research questions, and objectives.
- **Chapter 2** — Literature Review — This chapter reviews the literature on breast cancer screening, and deep learning applications to mammography.

- **Chapter 3** — Design — This chapter describes the design of the CNN architecture, and the data preprocessing pipeline.
- **Chapter 4** — Prototype — This chapter describes a basic implementation of the system, including experimental results, error analysis, and the discussion of potential improvements.

2. Background

Breast cancer screening may use multiple imaging modalities, including X-ray mammography, digital ultrasound, and hispathological microscopy, each offering different diagnostic information. Deep convolutional neural networks (CNNs) have proven to be remarkably effective across these modalities, by learning complex image features from a given dataset [Jia+24] [Car+24]. Widely-used mammography datasets include the Digital Database for Screening Mammography (DDSM) [Car+24] and the Curated Breast Imaging Subset of Digital Database for Screening Mammography (CBIS-DDSM) containing around 10,000 images [Car+24], and the INbreast dataset containing around 400 FFDM (full-field digital mammography) images [Car+24]. For histopathology, the BreakHis dataset, containing about 9,000 breast tissue images at 40x-400x magnification, and BACH (Breast Cancer Histology) dataset are commonly used as benchmarks [Jia+24] [Sri+23]. For ultrasound imaging, the BUSI (Breast Ultrasound Image) dataset, has also been used in CNN studies [Lat+24]. Such public repositories have enabled various CNN-based breast cancer detection and classification studies.

2.1. History

Computer-aided detection (CAD) systems for breast cancer originated in the 1980s with the advent of digital mammography, marking the beginning of using computer systems to assist radiologists. By the mid-1980s, major strides were made in developing algorithms for use in mammography (and also chest X-rays) [GCB08]. These early CAD systems used hand-engineered image processing, and statistical classifiers to detection calcifications or masses. In the 1990s, the first studies involving neural networks for CAD appeared. These involved the application of the first convolutional neural network to mammographic microcalcification detection [GCB08]. In subsequent years, the use of more traditional machine learning (ML) methods like support vector machines (SVMs), and decision trees were also applied to breast pathology, and imaging, using handcrafted features like texture, shape, and histogram statistics to classify regions as benign or malignant. Nowadays, CNNs are dominant in both mammography, and histopathology tasks, without using classical pipelines that extract features for a separate classifier. The use of CNNs allows for the direct learning of features from pixels, allowing for better accuracy, and generalisation [Ara+17]. Namely, studies have shown that CNN-extracted features used in an SVM achieved a 95.6% sensitivity on histology slides [Ara+17]. In general, CNN-based models outperform traditional feature-based SVMs or random forest classifiers across many breast image datasets, though SVMs, and trees are still used for certain tasks such as radiomics-based subtype classification [Guo+24]. The disadvantages of using CNNs usually include the necessity of large, labelled datasets, and heavy computational resources for their training, and inference. This however comes at the advantage of automated feature learning, and the identification of sub-

tle, complex patterns that classical machine learning models struggle to come to terms with. On the other hand, SVMs and decision trees can work with smaller datasets, but depend on domain-expert feature design, and may not generalise as well to unseen data [Ara+17].

2.2. Convolutional Neural Network Architectures, and Methods

Modern CNN-based architectures for breast cancer detection are typically reliant on established architectures (like ResNet, Inception, VGG, DenseNet, and EfficientNet) designed for medical imaging. Transfer learning is a common approach, where pre-trained models are fine-tuned on medical scans to compensate for limited annotated data [Sri+23]. This approach has been shown to outperform using CNNs as fixed feature extractors. Data augmentation like flips, rotations, colour jittering, and GAN-based augmentation is also commonly used, particularly in ultrasound, and pathology where datasets are limited [Lat+24]. Namely, Gupta et al. state that the random flipping, and rotation of ultrasound images, helps to overcome class imbalance on the BUSI dataset [Lat+24]. Explainability methods like Grad-CAM heatmaps are incorporated to visualise salient regions, and build trust in the model [Lat+24] [GJ24]. In segmentation tasks (like in cases of lesions), encoder-decoder networks like U-Net remain the industry standard, yielding critical performance on medical image segmentation tasks [Jia+24].

2.3. Mammography-based detection

CNNs have demonstrated a high diagnostic accuracy by classifying whole images or localised patches of mammograms. Studies often use the DDSM or INbreast dataset, and the Digital Breast Tomosynthesis (DBT) dataset. Namely, a commercial deep CNN-based system, trained on about 1,000 mammograms reached an AUC score of 0.82, comparable to an expert radiologist [Car+24]. Retrospective evaluations indicate the Computer Aided Diagnosis (CAD) systems powered by AI can improve radiologist sensitivity, such that their cancer detection rate rose from 51% to 62% after the introduction of AI aid as evidenced by Watanabe et al. [Car+24]. In a similar study, Kim et al. reports that an unassisted AI-CAD system achieved an AUC of 0.94, outperforming the radiologist's AUC of 0.88 [Car+24]. Akselrod-Ballin et al. achieved an AUC of 0.91 using a large linked dataset of mammograms, and patient records [Car+24]. In large screening challenges (with around 85,000 training images, and 68,000 external images), top CNN ensembles reached an AUC of 0.90 on held-out mammograms [Car+24]. Although still below expert performance, combining radiologists with AI yielded an AUC of 0.94, indicating the performance gains experienced when the two are synergised. CNNs have also been applied to lesion detection, and segmentation, with a particular case in which a U-Net-like model, trained on a CBIS-DDSM dataset, detected masses with 95.7% sensitivity, and a Dice score of 74.5% [Jia+24].

Popular CNN models for mammography include ResNet50, VGG16/19, Inception, and EfficientNet. For tasks like the detection of mass classification or microcalcification detection, such networks often outperform more traditional, feature-based

methods [Wan24] [Car+24]. Namely, Shen et al. found a fine-tuned CNN achieved an 88% accuracy whilst classifying mammographic tumourss, outperforming radiologists which achieved 83% accuracy [Wan24]. In another study, Yala et al. reports CNN risk models with an AUC of 0.84 against an AUC of 0.77 for radiologists [Wan24]. Object detection models like Faster-RCNN, and RetinaNet were also used within the Agarwal et al. study, which achieved a 99% accuracy for malignant-mass detection on the INbreast dataset, when applying Faster-RCNN to full-field mammograms [Jia+24]. More recent works, use transformer-based models using a pretrained vision transformer, and segmentation, which reaches a 99.96% accuracy on INbreast, by first extracting a mass ROI [Kum+24]. All in all, CNNs on mammograms achieve a high AUC value, often similar to or exceeding the performance of an average radiologist [Car+24].

2.4. Histopathology-based detection

CNNs have also been applied to histopathology slides for cancer classification. The BreakHis dataset (containing microscopic tissue patches at 40x-400x magnification) has been the de facto benchmark for this. Basic CNNs like ResNet, and VGG achieve more than 90% accuracy on binary classification tasks upon the BreakHis dataset (when classifying benign against malignant cancers) [Jia+24]. In Han et al. a CNN was applied to the BreakHis dataset, achieving a 93.2% accuracy [Jia+24]. More complex architectures can improve performance significantly more, such as in Munikoti et al. which combines a CNN with an LSTM (utilising ImageNet transfer learning), and achieving a 99% accuracy on binary classification in the BreakHis dataset [Sri+23]. Such deep learning models significantly outperform more traditional machine learning methods like Rao et al. which found that CNNs (with transfer learning) superseded SVMs or random forests on histology data [Yus+23]. Apart from patch classification, CNNs like U-net are used to segment nuclei or tumor regions to help analysis [Jia+24]. Modern approaches use attention or multiscale to capture complex tissue contexts [Sri+23]. In conclusion, CNNs on histopathology often achieve very high accuracies of 90-99% on curated datasets [Sri+23] [Yus+23], taking advantage of deep architectures, and large-scale patch augmentation.

2.5. Ultrasound-based detection

When breast tissue is dense, ultrasound imaging is often used as an imaging modality. The use of this, posits issues relating to speckle noise, and operator variability. CNNs have shown remarkable results despite this. Since public ultrasound datasets are smaller, transfer learning from networks like VGG-16, VGG-19, and AlexNet are often used. Wang et al. applied a multi-view CNN with ImageNet pretraining to a breast ultrasound, and achieved an AUC of 0.9468 [Wan24]. In another study, Gupta et al. fine-tuned EfficientNet-B7 on the BUSI dataset, and reported a 99.1% accuracy [Lat+24]. Augmentation methods like rotations, and colour jitter were used, and Grad-CAM explanation was used to focus on lesion features [Lat+24]. In addition, Rahman et al. deduced that a fine-tuned model of InceptionV3 outperformed VGG-19 for ultrasound tumour classification [ADC21]. In general, ultrasound CNNs commonly exceed 90% accuracy, with transfer learning, and data

augmentation necessary to compensate for limited sample sizes.

2.6. Transfer Learning, Data Augmentation, and Explainability

Since annotated breast imaging data is limited, transfer learning, and augmentation are essential. Almost all recent studies initialise CNNs with ImageNet weights, and fine-tune on medical images [Sri+23]. Fine-tuning routinely outperforms training from scratch or using CNNs as pure feature extractors. Strategies for augmentation, including geometric transforms, colour perturbations, and GAN-based synthesis help to ease overfitting problems. Namely, EfficientNet-based classifiers for ultrasound used random flipping, rotation, and colour jitter to improve minority malignant cases [Lat+24]. Other histology studies have also used flips, rotations, and staining augmentation to improve the diversity of tissue patches [Sri+23].

Explainable AI is becoming increasingly important in medical imaging, where methods such as heatmaps (like Grad-CAM, and saliency maps) are applied to highlight image regions to make predictions [Lat+24] [GJ24]. This helps to verify that CNNs focus on tumours rather than artifacts. Studies report overlaying class activation maps on mammographs or ultrasound to show ROI localisation. Namely, EfficientNet-B7 visualised Grad-CAM heatmaps on ultrasound to confirm lesion focus [Lat+24]. Explainability is also explored via feature attribution, and partial dependence, but Grad-CAM remains popular [Lat+24] [GJ24].

2.7. Performance vs. Traditional Methods, and Radiologists

CNN-based systems have been shown to outperform classical machine learning algorithms, and approximate expert-level analysis. Deep models have reduced false positives relative to older CAD systems, and improved sensitivity. On mammography tasks, CNNs matched or superseded radiologist sensitivity as demonstrated by Becker et al., which reported an equal AUC of 0.82 for a commercial CNN, and human experts [Car+24]. In larger trials, deep learning models achieved an AUC of 0.86 to 0.94, while experts scored slightly better, but, when combining AI with human experts, recall rates were consistently reduced [Car+24]. In ultrasound, CNNs have outperformed conventional classifiers like SVMs, and decision trees when given enough data [Yus+23]. In histopathology, CNNs significantly outperform custom feature methods, where hybrid CNN and TL models reached an accuracy of 97% on BreakHis, and classical machine learning models rarely exceeded 90% on the same task [Yus+23]. Thus, studies indicate that CNNs offer excellent sensitivity, and specificity, across different modalities [Car+24] [Wan24], demonstrating their practical utility within CAD systems.

3. Design

Breast cancer is a leading cause of death amongst women worldwide, and early detection is paramount to improve the chances of survival. Computer-aided detection (CAD) systems have long been used to assist in screening, and diagnosis workflows for long, but recent developments in convolutional neural networks (CNNs) have significantly boosted the rates of detection on natural image benchmarks. The aim of this project is to leverage state-of-the-art CNN architectures to build a decision support system for breast cancer detection in mammograms. The Digital Database for Screening Mammography (DDSM) will be used to train, and evaluate a CNN model, and the results will be used to indicate the presence of malignant tumours. Thus, the primary objective is to develop, and evaluate a CNN, that matches or exceeds the single-reader accuracy commonly achieved by radiologists in large-scale screening programmes. A working inference API will be developed, where the user may upload an image, run model inference, and receive a prediction.

3.1. Domain Context, and User Analysis

Mammography screening programs capture images of the breast, and use them to detect early signs of breast cancer. Radiologists are responsible for viewing multiple images per week. In this context, false negatives lead to delayed diagnoses that may pose serious health consequences, whilst false positives impose unnecessary stress on patients, and the system itself, with procedures like biopsies, and follow-up imaging creating an operational burden. Therefore, any AI-powered decision support system must prioritise very high sensitivity while maintaining a reasonable specificity to avoid recalls.

The primary users of this system will be professional radiologists, who will utilise suggestions from it, and clinicians who monitor screening quality, and throughput values. Radiologists will require not only malignancy scores, but also explainable outputs demonstrating where the model has detected abnormalities for validation purposes, and to manage trust issues in the system. Many published works lack sufficient end-to-end system design, and evaluation. By concentrating on both model performance, and system usability, this project fills a gap in showing a deployable prototype complete with explainable outputs, and inference benchmarks.

3.2. System Architecture, and Design

The resultant system will be comprised of a web application containing the following components: data preprocessing, model training, and evaluation, explainability, inference API, and user interface.

- **Data Preprocessing** — The DDSM dataset will be appropriately preprocessed to ensure that the images are in a suitable format for the model. This

will include resizing, normalisation, and augmentation of the images to improve the model's performance. Thus, horizontal flipping, rotations, and contrast adjustments may be applied on the fly to increase the diversity of the training data, and reduce overfitting. The dataset will also be split into training, validation, and test sets to ensure equal class balance across malignant, and benign classes.

- **Model Training, and Evaluation** — A CNN model will be trained on the DDSM dataset, being constructed with TensorFlow, and Keras. The model will then be evaluated using metrics like accuracy, precision, recall, and F1-score.
- **Explainability** — Grad-CAM will be used to overlay heatmaps on input mammograms, demonstrating various regions that drive the model's prediction.
- **Inference, and User Interface** — A Django-powered web server will be used for REST APIs, and hosting the user interface for users to interact with the system.

3.3. Technologies, and Methods

In light of the previous components, mature, and well-supported libraries were selected to ensure that the system is easy to maintain, and extend.

- **Data Handling** — Python as the fundamental language of choice for the entire stack, NumPy, and Pandas for data manipulation tasks, and OpenCV for image processing tasks.
- **Deep Learning** — TensorFlow for neural networks, with Keras frontend for ease of use, and fast prototyping.
- **Hyperparameter Tuning** — Optuna for automated, and efficient hyperparameter tuning, and model selection.
- **Explainability** — Grad-CAM through tf-keras-vis for visualising the model's attention on the input image, and highlighting the areas of interest.
- **API** — Django for high performance, and OpenAPI-compatible REST APIs.
- **Deployment** — Docker containers for easy deployment, and integration with cloud services (like AWS Elastic Container Service).

3.4. Model

The core of the system is a deep feature extractor powered by a convolutional neural network (CNN). Multiple methods will be explored, including alternative architectures like vision transformers (ViTs), but initially, a VGG16 backbone was selected. This provides proven performance on breast image tasks, and manageable complexity [FS25]. VGG16's uniform 16-layer architecture adapts well via transfer

learning from ImageNet, with prior work achieving a high accuracy on the BreakHis histopathology dataset by fine-tuning VGG16 [FS25]. Additionally, VGG16 has shown excellent generalisation on unbalanced data [FS25], making it suitable for the DDSM dataset's class imbalance.

3.5. Plan of Evaluation

The project as a whole will be evaluated using a variety of metrics. For the model itself, the most critical part of the system, the standard evaluation metrics of accuracy, precision, recall, and F1-score will be used to assess its performance on the DDSM dataset, including the Area Under the Receiver Operating Characteristic Curve (AUC). Together with Grad-CAM heatmaps, which will show the model's attention on the input images, these metrics will provide a comprehensive outlook on the model's classification performance. Since not much emphasis will be placed on the user interface, the evaluation will focus on the model's performance, and the explainability of the outputs. The inference API will be tested for usability, and performance, ensuring that it can handle multiple requests efficiently, and provide accurate predictions in a timely manner.

4. Prototype

The prototype for this project is an end-to-end image classification pipeline for distinguishing two classes of mammographic scans using deep learning. Transfer learning is applied, with a pre-trained convolutional neural network being used for the classification of scans. The system ingests data from TFRecord files, originating from the DDSM dataset. These are then preprocessed for the model to use as input. The latter is then trained, and evaluated on this data.
The following is a summary of the prototype's data ingestion, and preprocessing pipelines.

- **TFRecord Parsing** — A set of serialised examples from the DDSM dataset is parsed using TensorFlow's TFRecordDataset. Each example consists of three features: an image, a class label, and a normalised label.
- **Shuffling** — To improve stability in training, the dataset is shuffled with a buffer size of 31,000, and cached in memory for faster access across epochs.
- **Decoding, and Resizing** — Raw image bytes are kept in a unit8 tensor, and reshaped into a 299x299 image. OpenCV then resizes each image to 224x224 pixels, and replicates the single channel across three channels to form an RGB image, matching the requirements of the pre-trained model.
- **Dataset Conversion** — Images, and labels are converted into NumPy arrays X, and y for compatibility with Keras model training. Stratified splits are performed to split the dataset into training, validation, and test sets to maintain class balance.

The following details the CNN model's architecture, and definition.

- **Basis** — The VGG16 architecture is used as a basis for the model, which is pretrained on ImageNet as the fixed feature extractor without the top classification layers. All convolutional layers of the VGG16 basis are non-trainable to retain the learned low-level, and mid-level features.
- **Custom Classifier Head** — On top of the frozen basis, a series of layers is added to form a custom classifier head. This consists of layers like dropout layers to manage overfitting, batch normalisation layers to improve training speed, and a dense layer with Glorot uniform initialisation, and a ReLU activation function for the final classification. The final output layer involves a sigmoid activation function for binary classification.
- **Compilation** — The final model is compiled with the Adam optimiser, binary cross-entropy loss, and accuracy as the prime metric.

```
base_model = VGG16(
  weights="imagenet", include_top=False, input_shape=(IMAGE_SIZE,
    IMAGE_SIZE, 3)
)  # Base model VGG16 pretrained on ImageNet
base_model.trainable = False

""" Custom classification head """
model = Sequential(
  [
      base_model,
      Dropout(0.5),  # Dropout layers to prevent overfitting
      Flatten(),
      BatchNormalization(),  # Stabilises learning
      Dense(
    1024, kernel_initializer=GlorotUniform()
      ),  # Learn decision boundaries
      BatchNormalization(),
      Activation("relu"),
      Dropout(0.5),
      Dense(512, kernel_initializer=GlorotUniform()),
      BatchNormalization(),
      Activation("relu"),
      Dropout(0.5),
      Dense(1, activation="sigmoid"),  # Output layer for binary
    classification
  ]
)

model.compile(
  optimizer="adam", loss="binary_crossentropy", metrics=["accuracy"]
)  # Compile the model with Adam optimizer, binary crossentropy loss,
    and accuracy metric
```

Listing 4.1: Prototype model definition.

Note that this prototype borrows inspiration from the following work *https://www.youtube.com/watch?v=mmGy_tZU3lc*.

4.1. Evaluation

The system was evaluated using a primarily quantitative performance metrices.

- **Accuracy** — The prototype achieved an accuracy of 0.89 in testing on a held-out test set.
- **Precision, Recall, and F1-score** — Using macro-averaging across the two classes, precision, recall, and F1-score were used to account for any class imbalance.
- **ROC AUC** — The receiver operating characteristic (ROC) curve, and the macro-averaged area under the curve (AUC) were determined to assess the model's discrimination threshold independently of any single decision threshold.

In terms of ROC analysis, the model achieved an AUC value of 0.89, demonstrating a solid separability between the two classes at various thresholds.

4.2. Discussion of Results

In general, the prototype demonstrates feasibility of the proposed classification approach. The frozen backbone makes it easier to train the model, and lessens computational cost, but limits the model's ability to adapt to domain-specific features. In addition, the custom classifier head learns to distinguish classes understandably well.

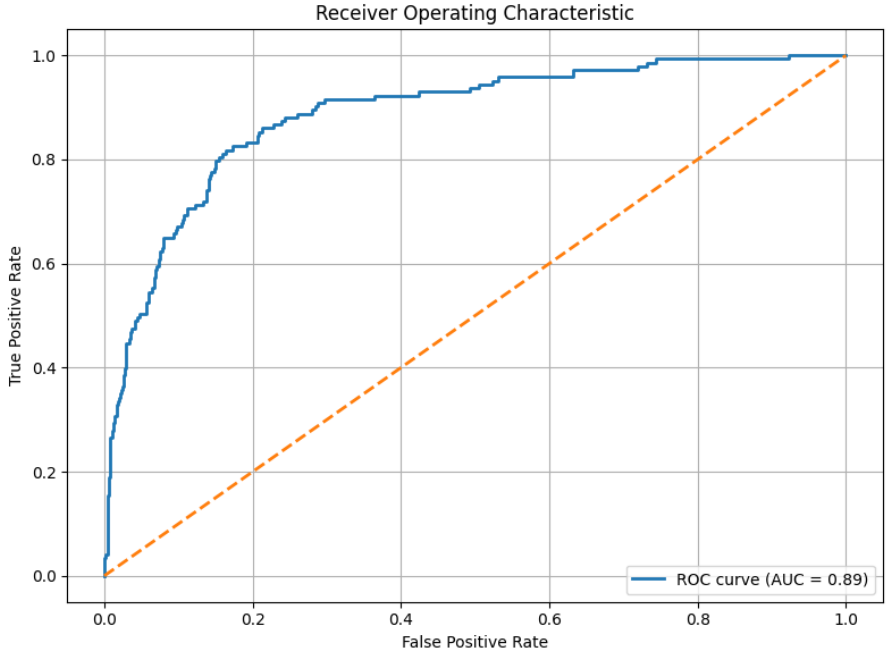

Figure 4.1.: Diagram of the ROC curve for the prototype model.

- Rapid prototyping using transfer learning enabled end-to-end functionality rapidly.
- Quantitative metrics like accuracy, precision, recall, F1-score, and AUC indicate sufficient performance for the initial project.

Metric	Value
Accuracy	0.89
Precision	0.55
Recall	0.61
F1-score	0.58
AUC	0.89

Table 4.1.: Quantitative performance metrics of the prototype model.

Note that these results were achieved with the limited training parameters of 1 epoch, and a batch size of 10. The performance of this model is expected to improve with further training, and optimisation.

4.3. Future Work

This prototype serves as a basic proof-of-concept, and a foundation for future work. The final system will require further refinement, and optimisation, as well as additional features, and functionality to make it more user-friendly, and robust.

- **Model Optimisation** — The model can be further optimised by adjusting its architecture, and increasing the number of epochs.
- **Advanced Architectures** — Alternative architectures could be experimented with like ResNet50, and EfficientNet that may offer better performance.
- **Data Augmentation** — tf.image operations like random flips, rotations, and brightness adjustments could be introduced to improve generalisation.
- **User Interface** — A user-friendly interface can be developed to allow users to easily upload images, and view results.
- **Deployment** — The model can be deployed as a web service through REST APIs, and be inferenced through them for user-friendly accessibility, particularly for clinical settings.
- **CI/CD** — Implementing continuous integration, and continuous deployment (CI/CD) practices such as the introduction of containerisation with Docker will ensure that the system is easily maintainable, scalable, and transferable across different platforms.

4.4. Approximate Plan of Work

The following is an approximate plan of work for the next stages of the project. It is subject to change, depending on the progress, and exigencies of it.

Task	**Description**	**Approx. Duration**	**Start Date**	**End Date**
Model Optimisation	Tune hyperparameters, increase epochs, and batch size	2 weeks	Jun 16, 2025	Jun 27, 2025
Advanced Architectures	Experiment with ResNet50, EfficientNet, or ViTs	3 weeks	Jun 30, 2025	Jul 18, 2025
User Interface Development	Build web UI for image uploads, and results together with endpoints	2 weeks	Jul 21, 2025	Aug 1, 2025
Deployment via REST API	Finalise Django REST API backend	2 weeks	Aug 4, 2025	Aug 18, 2025
CI/CD Integration	Containerise with Docker	2 weeks	Aug 19, 2025	Aug 22, 2025

Table 4.2.: Approximate Planned Timeline for the Project

5. Implementation

This chapter outlines the technical implementation of the breast cancer detection system, with a focus on data handling, model architecture, training methodology, and explainability. The system was implemented using TensorFlow, and Keras, running atop the Python interpreter. The data ingestion, and preprocessing steps will be covered alongside the architecture, and training of the convolutional neural networks used. Strategies to improve generalisation will be detailed, as well as the explainability module, and capturing, and visualising the results. Snippets of code are shown to demonstrate key points, with the full listings available in the appendix.

5.1. Setup

The necessary libraries are first imported, and the necessary helper functions are defined at the beginning of the notebook, to be used throughout. Namely, a garbage collection function is defined to clear as much memory as possible to prevent the system from crashing. This is called throughout the notebook. The function is called first to set the stage for the memory to be occupied solely by the subroutines in the following stages of the implementation.

5.2. Data Manipulation

This section focuses on loading, and preparing data for training, and testing from TFRecord files. A random seed is firstly set so that results are reproducibile. Due to the limitations of the hardware utilised for this project, a small subset of the DDSM dataset, sourced from Kaggle, was utilised, with pre-augmented images. The dataset was provided in the form of TensorFlow's TFRecord format, which is a binary format that allows for efficient storage and retrieval of large datasets. Each TFRecord file contains three fields, `image`, `label`, and `label_normal`. The `image` field contains the raw, flattened image data, the `label` field contains the label for the image in the form of a binary integer class (0 for benign, and 1 for malignant classifications), and the `label_normal` field contains a normalised version of the label for direct use in training.

Two empty lists are first initialised for images, and labels with a dictionary defined that specifies the structure of TFRecord data. A function is then defined to process the TFRecord files, and store the results into the previously defined `images`, and `labels` lists.

```
def read_file(files):
    mydata = (
        tf.data.TFRecordDataset(files, num_parallel_reads=4)
        .shuffle(buffer_size=10000)
        .cache()
    )
```

```
    mydata = mydata.map(
        lambda x: tf.io.parse_example(x, features), num_parallel_calls
    =4
    )
    for image_features in mydata:
        image = tf.io.decode_raw(image_features["image"], tf.uint8)
        image = tf.reshape(image, [299, 299])
        image = np.asarray(image)
        images.append(image)
        labels.append(image_features["label"].numpy())
```

Listing 5.1: Python example

The garbage collection function is invoked before the files are parsed, to avoid any memory issues. The target TFRecord files are defined in a list, prior to being processed. The parsing function is finally called on the previously defined list.

5.2.1. Preparing Data for Training

This section will discuss the cells that process the already loaded training data to make it more understandable to the CNN later in the notebook. Each image is first saved as a `.jpg` file in the `train` directory with enumerated filenames. The list of labels is saved as a CSV file within the same directory. The labels are then loaded, and converted to boolean format, ideal for the binary classification task the CNN is to perform. The list of `.jpg` files from the `train` directory is retrieving using `glob`, with the numeric part of the filename extracted, and storing in the `mysplit` list for further sorting. A Pandas DataFrame is defined for training, with columns to store the image paths, and binary label, all sorted by their numeric value.

5.2.2. Preparing Data for Testing

This section details methods similar to those seen within the previous section, but more suited towards preparing testing data. The following cell begins by loading test images, and labels from NumPy files, and concatenates them to form `x_test` for images, and `y_test` for labels. The `y_test` array is then converted to boolean format as was done for training labels. The testing data is then saved as `.jpg` files within the `test` directory with enumerated filenames. The test labels are then loaded from the CSV file, and casted as strings. Similar to the previous section, the filenames are retrieved as a list of `.jpg` files using `glob`, with the numeric part of the filename extracted via `glob`. A Pandas DataFrame is finally created with this information.

5.2.3. Visualisation

The dataset is visualised within this section, with the garbage collection function being called first to alleviate the system's memory as much as possible.

5.2.4. Data Generators

The data processed in the previous sections is finally encapsulated into generators to feed it into the model during training, and testing. An `ImageDataGenerator`

object is first defined to preprocess images by rescaling their pixel values. Each pixel is normalised into the range 0, to 255 by being divided by 255. This is known to improve the model's training efficiency, and performance. The training generator yields batches of preprocessed training images, and their corresponding labels from the `traindf` DataFrame. This feeds the model during training. A similar generator is made for the testing set.

5.3. Model Definition

The model used for binary classification of breast tissue is a convolutional neural network, based upon a frozen `VGG16` layer, that uses pre-trained `ImageNet` weights. This is then followed by `Dense`, and `GlobalAveragePooling2D` layers, and fine-tuned accordingly.

```
myconv = VGG16(weights="imagenet", include_top=False, input_shape=(299,
    299, 3))
x = myconv.output
x = GlobalAveragePooling2D()(x)
x = Dense(1, activation="sigmoid", name="classifier")(x)
model = Model(myconv.input, x)
```

Listing 5.2: Python example

5.3.1. Training the Model

The model will now be trained, based on the previous definition. Garbage collection is called prior to training, to avoid memory issues. Important callbacks are first defined to stop training if parameters like validation loss do not improve, and for the model with best validation accuracy to be saved. The model is compiled with an `adagrad` optimisation function, and binary cross-entropy loss function. The model is finally trained, with data being fed from the previously defined generators.

5.3.2. Evaluating the Model

This section evaluates the previously trained model using standard metrics like accuracy, and loss on the training, and validation sets. In the following cell, these metrics are extracted into variables for later use from the performance of the model across different epochs during training. This is then plotted, to visualise how the model's accuracy, and loss scores improved. Predictions are made on the test set are made in order to generate a final classification report, and confusion matrix. The previously-made predictions, and actual values are gathered, and fed into `scikit-learn` to generate the confusion matrix. The results are then plotted using `seaborn`. The same values are used to generate the final classification report.

5.4. Model Explanation

This final section touches upon XAI or explainable AI, by implementing *Grad-CAM* through the `tf-explain` module. This provides a visual indication of the parts of the image that influenced the classification made of the model, explaining the results

achieved. This is done for the first ten images within the testing set, but can easily be done for any other image as well. The following function accepts a path to an image, the model used, and it's final layer, and returns the explained version of the image, and the classification result. A series of images is then passed through the aforementioned function, and plotted into a singular figure. In the above plot, warmer colours indicate higher influence on the model's predicted class, and cooler colours the opposite.

A. Appendix Title

The code listing for the project prototype.

Bibliography

[ADC21] G Ayana, K Dese, and SW Choe. *Transfer learning in breast cancer diagnoses via ultrasound imaging. Cancers 13: 738*. 2021.

[Ara+17] Teresa Araújo et al. "Classification of breast cancer histology images using convolutional neural networks". In: *PloS one* 12.6 (2017), e0177544.

[Car+24] Alessandro Carriero et al. "Deep learning in breast cancer imaging: State of the art and recent advancements in early 2024". In: *Diagnostics* 14.8 (2024), p. 848.

[FS25] Tanjim Fatima and Hamdy Soliman. "Application of VGG16 Transfer Learning for Breast Cancer Detection". In: *Information* 16.3 (2025), p. 227.

[GCB08] Maryellen L. Giger, Heang-Ping Chan, and John Boone. "Anniversary Paper: History and status of CAD and quantitative image analysis: The role of Medical Physics and AAPM". In: *Medical Physics* 35.12 (2008), pp. 5799–5820. DOI: https://doi.org/10.1118/1.3013555. eprint: https://aapm.onlinelibrary.wiley.com/doi/pdf/10.1118/1.3013555. URL: https://aapm.onlinelibrary.wiley.com/doi/abs/10.1118/1.3013555.

[GJ24] Daraje kaba Gurmessa and Worku Jimma. "Explainable machine learning for breast cancer diagnosis from mammography and ultrasound images: a systematic review". In: *BMJ Health & Care Informatics* 31.1 (2024), e100954.

[Guo+24] Ya Guo et al. "Machine learning and new insights for breast cancer diagnosis". In: *Journal of International Medical Research* 52.4 (2024), p. 03000605241237867.

[Jia+24] Bitao Jiang et al. "Deep learning applications in breast cancer histopathological imaging: diagnosis, treatment, and prognosis". In: *Breast Cancer Research* 26.1 (2024), p. 137.

[Kum+24] Dip Kumar Saha et al. "Segmentation for mammography classification utilizing deep convolutional neural network". In: *BMC Medical Imaging* 24.1 (2024), p. 334.

[Lat+24] M Latha et al. "Revolutionizing breast ultrasound diagnostics with EfficientNet-B7 and Explainable AI". In: *BMC Medical Imaging* 24.1 (2024), p. 230.

[Sri+23] Mahati Munikoti Srikantamurthy et al. "Classification of benign and malignant subtypes of breast cancer histopathology imaging using hybrid CNN-LSTM based transfer learning". In: *BMC Medical Imaging* 23.1 (2023), p. 19.

[Wan24] Lulu Wang. "Mammography with deep learning for breast cancer detection". In: *Frontiers in oncology* 14 (2024), p. 1281922.

[Yus+23] Marina Yusoff et al. "Accuracy analysis of deep learning methods in breast cancer classification: A structured review". In: *Diagnostics* 13.4 (2023), p. 683.