TU Dublin, Tallaght Campus

BSc (Hons) in Computing with Machine Learning and Artificial Intelligence

Comparing Pre-trained CNNs for Malignant vs. Benign Mammogram Classification

Peter Murphy
Department of Computing

Supervised by
Musfira Jilani
Department of Computing

April 27, 2025

Declaration

I hereby certify that the material, which I now submit for assessment on the programmes of study leading to the award of Bachelor of Science (Hons), is entirely my own work and has not been taken from the work of others except to the extent that such work has been cited and acknowledged within the text of my own work. No portion of the work contained in this thesis has been submitted in support of an application for another degree or qualification to this or any other institution.

Peter Murphy

April 27 2025

Acknowledgements

I would like to thank Musfira Julani for her patience, support, and encouragement throughout this project and for helping me begin my machine learning journey with the Applied Machine Learning module.

I would like to express my gratitude to Zaur Gouliev for his coherent and interactive teaching of deep learning and convolutional neural networks.

Finally, I would like to thank my mother for her support and for allowing me to verbalize my conclusions before committing them to this project.

List of Figures

Contents

Abstract

This study investigates the performance of pre-trained Convolutional Neural Networks (CNNs) for classifying malignant vs. benign mammograms, using the benchmark VinDr-Mammo dataset. The models tested include VGG-16, ResNet-18, and MobileNetV2, each optimized and fine-tuned through transfer learning on the dataset. Appropriate performance metrics were selected for the medical context of this study. The study also addresses some major challenges with medical image datasets, such as class imbalance and the high computational cost associated with training deep CNNs on large datasets. Although no great accuracy was achieved from any of the models in this study, some valuable insights were acquired into a potential direction for medical image analysis with limited resources. The conclusions of this study are informed by both ethical analysis and measured results.

Introduction

Breast cancer continues to cause major public health challenges worldwide as one of the most common cancers found in women. In 2022 alone, breast cancer caused 670,000 deaths globally, underscoring the urgency and necessity of effective diagnostic techniques *(World Health Organization, 2024)*. As with many forms of cancer, the survival rate of breast cancer patients is dramatically increased with early detection. Early detection not only increases patient survival but also significantly reduces the need for invasive treatments such as lumpectomies and mastectomies.

Machine learning has been used in cancer detection for almost 20 years, with a growing body of research demonstrating its effectiveness in breast cancer detection. Recent studies, such as that by Asri et al. *(2016)*, have shown machine learning techniques to be effective in detecting breast cancer in its early stages. This study aims to investigate further the accuracy and effectiveness of modern machine learning models, specifically focusing on the comparative performance of pre-trained convolutional neural networks (CNNs), utilising a recently available dataset.

The VinDr-Mammo dataset was released in 2022 by the Institutional Review Board of Hanoi Medical University Hospital and Hospital 108, where the data was collected and ethically approved. The dataset consists of 20,000 images from 5,000 mammography exams undertaken between 2018 and 2020 *(Pham, H. H et al., 2022)*. The images have been pre-annotated by radiographers using the BI-RADS standard for reporting breast imaging findings. BI-RADS classifications indicate the likelihood of malignancy and have been annotated for both the breast and any findings that were flagged by the radiographers.

This thesis will apply transfer learning techniques to VGG-16, ResNet-18, and MobileNetV2 architectures using VinDr-Mammo. Before the models can be trained on the raw dataset, we will first apply standard image pre-processing techniques to ensure consistency of size and resolution, such as normalization and histogram matching. The models will then be fine-tuned through transfer learning, adjusting their weights to optimise their predictive performance in classifying breast anomalies. Transfer learning is a machine learning technique in which models that have been trained for a particular task are applied to and optimised for a related task *(Murel, J. and Kavlakoglu, E., 2024).* VGG-16, ResNet-18, and MobileNetV2 have been pre-trained for image classification tasks and will be optimised for the more specific task of breast cancer classification.

Literature Review

In preparation for this study's methodology, a review of the relevant literature will be conducted. The review will focus on the associated dataset, the technologies utilized, and the techniques employed for breast cancer detection. This review will investigate key studies and technologies relevant to this project.

Carrilero-Mardones et al. (2024) proposed a model for BI-RADS classification of breast tumors using deep learning techniques. Their study focuses on classifying breast ultrasound images based on BI-RADS (Breast Imaging-Reporting and Data System) terminology. They found that classifying BI-RADS 4A or lower as benign and other categories as malignant yielded effective results. This approach, which simplifies classification into binary categories, is replicated in the approach used in this study, where malignant and benign classes are identified. Additionally, the authors noted that despite its deeper architecture, VGG19 performed similarly to VGG16 but with a higher computational expense. Given these findings, this study will opt to use VGG16 to balance performance with efficiency, as it has been shown to provide strong results in image classification tasks without the added complexity of deeper models.

Regarding recent applications of CNN-based models in mammography, Wong et al. (2020) reported that modern deep CNN systems can detect breast lesions with high accuracy, in some instances approaching the performance of radiologists. Advancements such as these illustrate the potential of CNNs in medical image analysis.

It is imperative that the correct performance metrics are selected for this study Asri et al. (2016) utilised precision, accuracy, recall and F1-score in their analysis. Recall is a critical metric to be considered in a medical context in order to minimize false negatives. When detecting cancer it would be reasonable to assume that false negatives will be more harmful than false positives. Precision and accuracy will also be measured. When measuring performance this study will encounter the impact of class imbalance in medical image datasets. Buda et al. (2018) performed a study on class imbalance in deep CNN classifiers and found that handling imbalance by oversampling minority classes consistently yields higher prediction accuracy than undersampling the majority class, particularly under severe imbalance ratios

When implementing prediction systems with limited resources Yang et al. (2021) emphasize the persistent challenges posed by medical imaging datasets, noting that models trained on limited or single-center data often fail to generalize to new clinical settings. They promote incorporating mutli-center data to improve breast cancer detection with machine learning.

Dataset Selection

The dataset selected for this study is the VinDr-Mammo dataset, which is a benchmark dataset in full-field digital mammography. This dataset was released to advance the development of computer-aided detection (CAD) systems for breast cancer diagnosis *(Nguyen, H.T. et al., 2022).*

I selected this dataset because of my strong personal interest in computer vision and its proven ability to solve real-world problems. Computer vision is one of the fastest-growing areas of research in machine learning, with many practical applications, such as self-driving cars, facial recognition for authentication, and medical diagnostics.

Among these applications, cancer detection stands out as an area where computer vision can significantly improve people's lives. The VinDr-Mammo dataset offers a new and exciting opportunity to optimize the techniques used in medical image analysis through machine learning, particularly in the context of breast cancer detection.

Data Source

The VinDr-Mammo dataset was released in 2022 by the Institutional Review Board of Hanoi Medical University Hospital (HMUH) and Hospital 108 (H108), where the data was collected and ethically approved. The dataset consists of 20,000 images from 5,000 mammography exams undertaken between 2018 and 2020 *(Pham, H. H et al., 2022)*. DICOM metadata accompanies the images, including image specifications, the patient's age, and the imaging device model. DICOM (Digital Imaging and Communications in Medicine) is the international standard to transmit, store, retrieve, print, process, and display medical imaging information *(DICOM, 2024).* All identifying patient information was removed to ensure privacy. Findings such as skin retractions and masses have been annotated in the images, and related information is stored in an accompanying csv file *(Pham, H. H et al., 2022).*

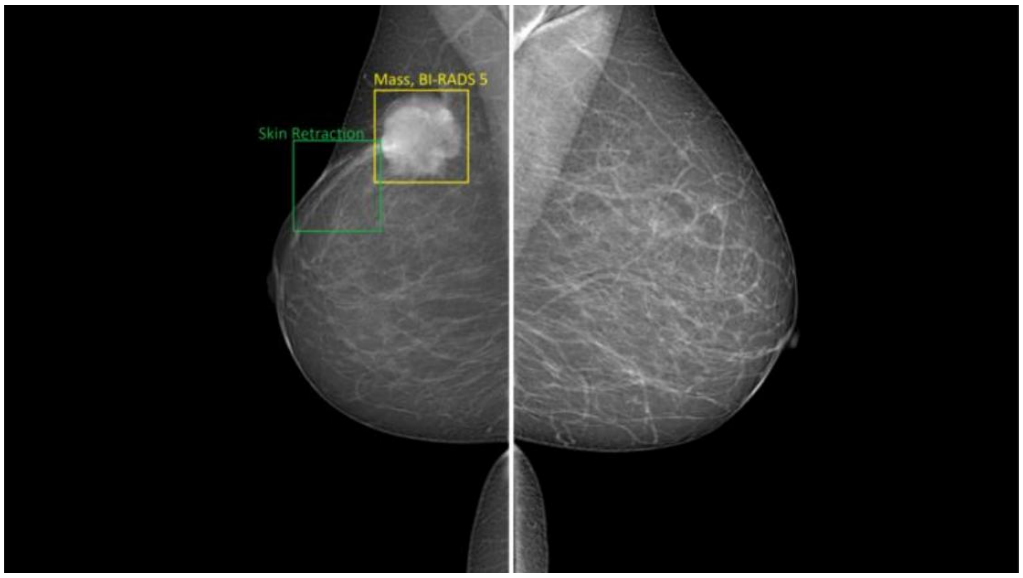

Figure 1: Mammogram example from the VinDr-Mammo dataset.

Note: The bounding boxes in the above image are not present in the training images. Their coordinates are simply specified in an accompanying csv.

VinDr-Mammo contains images from patients who have been assessed by radiographers and categorized under BI-RADS. BI-RADS or 'The Breast Imaging Reporting Data System' is a scoring system that doctors use to categorize the results of mammograms, breast ultrasounds, and breast MRIs. BI-RADS categorizes results from 0 to 6, indicating if benign or malignant cancer is likely and suggesting appropriate follow-up measures *(BreastCancer.org, 2024).* BI-RADS also defines a categorization system for breast density ranging from A (Mostly fatty; containing little fibrous tissue) to D (Extremely dense; includes a lot of fibrous tissue) *(BreastCancer.org, 2024).* Breast density is known to affect the visibility of lesions, which makes mammograms more difficult to interpret, both for radiographers and CNNs. This study will, however, include all categories of breast density in model training to ensure the best chance of accurate classification regardless of breast density.

Data Format

VinDr-Mammo includes three comma-separated values files containing further information relating to the examinations:

- **metadata.csv**
  Contains the DICOM information, including image specifications, patient age, and machine specifications.

- **breast-level_annotations.csv**
  Contains the BI-RADS assessment, the density of the associated breast for each image, and some image specifications.

- **finding_annotations.csv**
  This folder contains data relating to findings annotated in each image (e.g., skin retraction, mass, etc.). Each marked finding also includes a BI-RADS assessment.

Data Split

The dataset has been pre-split between 'Training' and 'Test' to provide consistency between studies. The chosen split can be found in both the breast-level_annotations and finding_annotations files.

Class Distribution

The following chart from the VinDr-Mammo release paper shows the distribution of BI-RADS classes (1-5) for both the training and test splits. The dataset contains 20,000 images in total, with two views per breast; that is, 10,000 breasts examined by radiographers and each assigned a BIRADS rating.

	Breast BI-RADS					Total
	1	2	3	4	5	
Training	5,362 (67.03%)	1,871 (23.39%)	372 (04.65%)	305 (03.81%)	90 (01.12%)	8,000
Test	1,341 (67.05%)	467 (23.35%)	93 (04.65%)	76 (03.80%)	23 (01.15%)	2,000
Overall	6,703 (67.03%)	2,338 (23.38%)	465 (04.65%)	381 (03.81%)	113 (01.13%)	10,000

Figure 2: Statistics of BI-RADS scores assigned by radiographers (Pham et al., 2022)

One of the main challenges when working with the VinDr-Mammo dataset is the significant class imbalance favoring benign cases over malignant cases. This is not an unknown problem when dealing with radiography images relating to cancer, since many of the images will have been scanned as a precaution and not because there were any certain signs of malignancy. This study will attempt to handle the class imbalance found in the dataset using techniques with proven results in the literature.

Dataset Collection

The VinDr-Mammo dataset can be downloaded from physionet.org. Although the dataset is preprepared for data science and machine learning applications, there are some adjustments that need to be made before it is fit for purpose. That includes converting the DICOM images to a format like JPG that is more suited for use with machine learning models. Converting from DICOM to JPG will also reduce the file size. The images can be batch converted using the PyDicom Python library *(PyDicom, 2023).* Images should be resized to ensure that they are all of a standard dimension. The images have been split and annotated as test or training, but the split should be analyzed to ensure that it has been stratified across important features like BIRADS.

Data Pre-processing

As with any machine learning study, data preprocessing is an important and necessary step to ensure that the raw data is converted to a usable format for the algorithms to process. This report will discuss the preprocessing steps, such as CLAHE (Contrast Limited Adaptive Histogram Equalization), image resizing, and normalization. By optimizing the quality of the images, we can ensure that fair comparisons will be made in the next phase of this study.

Convert the images from DICOM to PNG format.

The CNN models in this study do not support DICOM formatting, making conversion to PNG the first necessary step in data pre-processing. The script will iterate through the images and systematically convert them to .png files in a new folder. The script will also ensure that the original folder structure is preserved, to retain access as intended to the associated categorical data found in accompanying csv files. The images will be individually normalized as part of this process to stretch the range of pixel intensities, improving contrast and readability.

Resize the images

After converting the images, they should be resized for use with CNNs. Most models require consistent input dimensions, so resizing the images ensures that all of them are consistent in size. Standardizing the size also reduces the computational load and ensures the models can process the data efficiently. The images are resized to 256x256, which works well with most CNN models using `CV2.INTER_AREA` preserves image quality when downscaling through the use of pixel resampling.

Apply CLAHE contrast enhancement.

CLAHE (Contrast Limited Adaptive Histogram Equalization) considers the global contrast of the image. With CLAHE, the image is split into multiple sections or 'tiles', which are individually equalized to avoid darker tiles influencing lighter ones. It is particularly common with medical images to find large areas of low contrast. Using OpenCV's cv2 library, this equalization method can be easily applied to the images in this dataset. As with each step in this process, special care is also taken in order to preserve the original folder structure.

Before CLAHE:

Figure 3: Before CLAHE example

After CLAHE:

Figure 4: After CLAHE example

Apply Augmentations

Given the significant class imbalance in the VinDr-Mammo dataset, it stands to benefit from appropriate data augmentations to synthetically increase the representation of the underrepresented malignancy class. Data augmentations can be dynamically applied during training using the PyTorch Datasets `transforms` module. In this study, a range of augmentations were tested, including random flips, mild color jitter, and a light Gaussian blur. It was found that both color jitter and Gaussian blur negatively impacted performance, likely due to their adverse effect on the model's ability to detect finer edges. However, random flips led to a slight performance improvement, which is likely attributed to their role in addressing the class imbalance, especially when paired with the random weighted sampler that otherwise risks repeatedly loading the same image.

```
transforms.RandomHorizontalFlip(p=0.3),                # Included
transforms.RandomVerticalFlip(p=0.3),                  # Included
transforms.ColorJitter(brightness=0.1, contrast=0.1)   # Removed
transforms.GaussianBlur(kernel_size=3)                 # Removed
```

Crop to breast region using OTSU thresholding

Following some unsuccessful initial trainings of the models, a list of potential problems were identified, one being the background noise and large dark/light areas in the images. OTSU thresholding is a method for separating the background from the foreground.

Splitting the Dataset

In preparation for testing the CNN models against this dataset, the images should be split into those that will train the models and those that will test their performance. The VinDr-Mammo dataset has a pre-determined split indicated in the breast-level_annotations.csv. The dataset has been split into 1,000 test exams and 4,000 training exams, with the frequencies of each BI-RADS category, density level, and abnormality category being preserved by applying an iterative stratification algorithm *(Hieu T. Nguyen et al. "A large-scale benchmark dataset for computer-aided diagnosis in full-field digital mammography")*. This study will not attempt to improve on the stratification already achieved.

Please note that although dataset splitting was achieved in the data pre-processing phase of this study, the model implementation did not require it. This is because images are dynamically separated into training and testing data frames during data loading.

Model Development

The model development phase of this study focuses on selecting, implementing, and fine-tuning convolutional neural networks (CNNs) to classify breast anomalies using mammography images. At this phase, we introduce an open-source deep learning library called PyTorch, which supports the development and training of CNNs selected for this study. Fundamentally, PyTorch supports complex computations through the use of Tensors, which are multi-dimensional arrays similar to NumPy arrays but optimized for CUDA (Compute Unified Device Architecture) operations on GPUs. This greatly improves computational efficiency, which is particularly relevant when training CNN models on large datasets such as VinDr-Mammo.

Model Selection

Selecting appropriate models for this study is arguably as essential as their training and fair comparison. This study has chosen to focus only on convolutional neural networks, which have been proven to excel at image classification tasks. This is due to the convolutional layers that the architecture is named after. At the convolutional layer, structured data is broken down into sections, which allows the network to identify increasingly abstract features. This technique finds a perfect application with the structured pixel matrices of images, where CNNs can be used to detect edges, features, and objects. Ultimately, the models used in this study were selected for their alignment with the following criteria:

Proven performance with computer vision tasks
The primary condition that a model must satisfy is that it has proven its ability to accurately recognize patterns in images and classify images based on its findings. Although CNNs tend to be developed and optimized for computer vision tasks, there are architectures that are more suited to other ML tasks, such as natural language processing and autonomous vehicle driving.

Availability and accessibility
This study aims to advance the development of computer-aided breast cancer detection. For this reason, it is critical that the models used in their findings can be easily replicated and improved. Computational efficiency can also be a concern when considering the accessibility of models selected, particularly in a medical setting.

Suitability for medical applications
This study's ultimate goal is to aid radiographers and medical professionals in early breast cancer diagnosis. For this reason, it is critical that the models selected are suitable for such an important application. I am running a few minutes late; my previous meeting is running over.

Considering the criteria discussed above, the following models were selected.

VGG-16
Selected for its widespread availability and proven performance. Based on the paper 'Very Deep Convolutional Networks for Large-Scale Image Recognition' by Simonyan, K. & Zisserman, A. (2015), VGG is one of the deepest convolutional networks available. The depth of the model refers to the number of 'hidden', or weighted layers. There are a number of VGG model architectures available, which range between 11 and 19 layers. Simonyan, K. & Zisserman, A. (2015) proved in their paper that more layers translate to increased accuracy. However, the increased performance comes at the cost of computational efficiency. VGG-16 strikes an appropriate balance between performance and efficiency for the application of this study.

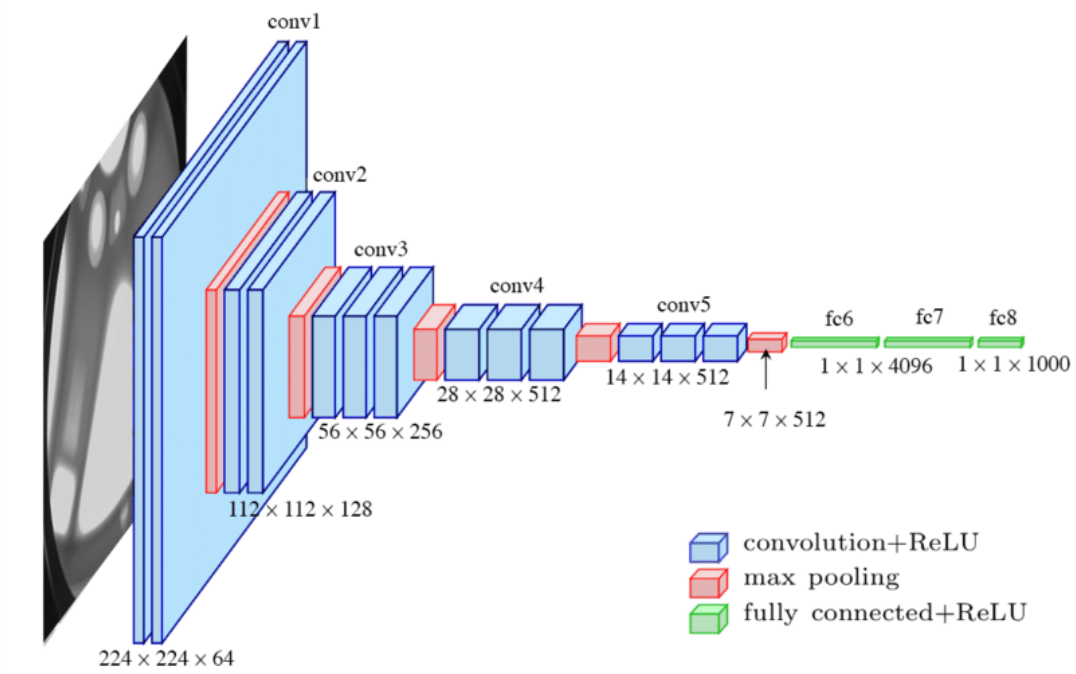

Figure 5: VGG Architecture (Simonyan & Zisserman, 2015)

ResNet-18
Another very deep model architecture with proven performance, ResNet won the ImageNet Large Scale Visual Recognition Challenge in 2015 (Russakovsky, O., Deng, J., Su, H. *et al.).* ResNet is known for its approach to the vanishing gradient problem through the use of residual blocks and skip connections. The problem occurs during the backpropagation phase of training and is, as such, particularly prominent with deeper neural networks. Having been based on a paper by He et al.

(2015) that aimed to ease the training of increasingly deeper networks, ResNet is relatively straightforward to implement. With 18 layers, ResNet-18 is a suitable candidate for comparison to the other models in this study.

MobileNetV2

Google developed the MobileNet CNN architecture as a lightweight CNN suitable for mobile applications. The models are highly efficient and aim to increase accuracy in resource-constrained environments. Their efficiency and accessibility make them important architectures to consider in the context of this study (Howard et al., 2017). MobileNetV2 builds upon the ideas of MobileNetV1 to achieve surprising accuracy with significantly lower training times.

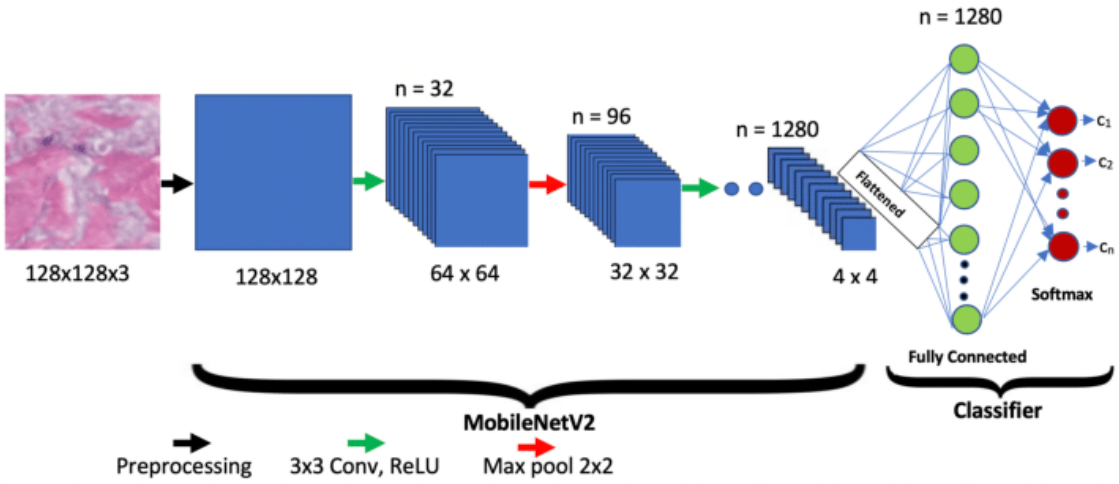

Figure 6: MobileNetV2 Architecture (Akay et al., 2021)

Dataset Preparation and Transformation

Although the images in this dataset have been pre-processed, additional preparations are required to ensure that the data is ready for training the models. Crucially, the target feature must be generated (malignant vs. benign) by mapping the BI-RAD values. Additionally, that dataset will also need to be contained within the structures expected by PyTorch. That means the data will need to be contained within a PyTorch Dataset, which in turn requires a DataLoader to pass the data to the model.. These steps will ensure an efficient and training process with minimum interruptions.

BI-RADS Binary Categorization: Malignant or Benign

Since the models will be trained for breast cancer detection, we must first decide on the type of classification that we intend to deliver. Upon reading the literature surrounding deep learning based breast cancer detection, a common approach is to

split the range of BIRADs into a binary classification problem of either malignant or benign. The BI-RAD categories are as follows:

0. Incomplete/ Additional imaging required
1. Negative
2. Benign (noncancerous) finding. 0% probability of malignancy.
3. Probably benign. < 2% probability of malignancy.
4. Suspicion of malignancy. 2-95% probability of malignancy.
5. Highly suggestive of malignancy. >95% probability of malignancy.
6. Known biopsy-proven malignancy.

Based on these categories and the probability of malignancy defined for each, we can map BI-RADS 1 and 2 to the "benign" class and BI- RADS 4, 5 and 6 to the "malignant" class as is recommended by Pham et al. in their 'VinDr-Mammo: A large-scale benchmark dataset for computer-aided diagnosis in full-field digital mammography' paper. This creates a clear binary classification problem that aligns with many breast cancer detection approaches in the literature. This mapping can be implemented in the Dataset's get method to update the breast_birads value as it's accessed dynamically.

```
 if birads_value in [1, 2]:  # Benign
    label= 0
else:  # Malignant (BI-RADS 3, 4, 5)
    label= 1
```

Define the Dataset

In order to effectively iterate through the dataset and feed batches of data into our model for training, we will need to wrap everything up in a custom PyTorch dataset. Our dataset will extend the `torch.utils.data.Dataset` class. By extending the Dataset, we can define how our data is stored and retrieved using our DataLoaders in the next step. This is crucial for ensuring that our data can be lazy loaded instead of being stored entirely in memory. The Dataset class contains three methods that the DataLoader requires for further processing.

- **__init__** is the constructor that initializes the Dataset's attributes.
- **__len__** returns the total number of samples – this is a requirement for the DataLoader.
- **__getitem__** returns one sample at a specific index. This includes the image itself and the label/BI-RADS value.

The Dataset also contains a transform attribute, which can be used to preprocess and augment data. They can be accessed in the `transforms` module of the `torchvision` library. PyTorch provides a variety of built-in transforms, such as

`transforms.ToTensor()` and `transforms.Normalize()`. Multiple transformations can be combined using `transforms.Compose()`, ensuring that the data is processed consistently before it is passed to the model. Using transforms enhances data variability, which improves model generalization. This study will apply the augmentations as specified in the data pre-processing section of this report.

Define the DataLoaders

We can use a PyTorch DataLoader from `torch.utils.data` to efficiently handle data during training and evaluation. The DataLoader wraps our custom Dataset and handles important tasks like batching, shuffling, and optional parallel loading. This approach lets us feed the model batches instead of storing the entire dataset in memory, which is essential for large datasets. A batch size of 32 was chosen as optimal for this study. Key parameters include:

- **batch_size:** Number of samples in each mini-batch.
- **shuffle:** Randomizes the order of samples for better generalization.
- **num_workers:** Allows parallel data loading if supported by the environment.

We will build DataLoaders for the training and validation sets for this project. Since the Vindr-Mammo dataset only provides two splits (training and test), the test set also serves as our validation set. A validation set is typically used during training to fine-tune model parameters, while a proper test set is reserved for final performance checks on unseen data. In this case, we use the same set for both validation and testing, which still allows us to measure performance and compare models.

Model Implementation Strategy

Model implementation begins with one of the selected model architectures from the PyTorch torchvision library. Using pre-trained weights accelerates training and allows us to leverage feature representations learned on large-scale image datasets like the ImageNet database provided by Stanford Vision Lab (2020). When initializing the model, the weights utilized can be defined with an optional parameter, or ImageNet weights are typically taken as the default.

In this binary classification approach, the final classification layer is replaced with a fully connected layer that outputs two classes (benign vs. malignant). The adaptation of the final classification layer is achieved through a built-in method that varies between models, although the principle remains the same.

The model is then moved onto the GPU to take advantage of CUDA (Compute Unified Device Architecture) parallel operations for more efficient computations.

We define our loss function as a Binary Cross-Entropy Loss function, which is a standard choice when solving binary classification problems. Adam is selected as the optimizer, with a learning rate of 1e-4. Adam converges more quickly and robustly than basic optimizers like vanilla SGD (Stochastic Gradient Descent), especially in convolutional networks. This learning rate is relatively small, which is advisable when retaining pre-trained weights.

VGG-16 Implementation

Following the implementation strategy as discussed, implementing a VGG-16 model begins with importing `vgg16` from the `torchvision.models` package. The model can then be initialized using pre-trained weights from the ImageNet database.

```
From torchvision.models import vgg16, VGG16_Weights
vgg16 =vgg16(weights=VGG16_Weights.IMAGENET1K_V1)
```

To adapt the final layer for binary classification, we can reference it using the `classifier` module, where `6` is the index of the final fully connected layer. Since our problem has a defined number of inputs and outputs, we can update the classifier to a PyTorch linear transformation layer with two output features.

```
vgg16.classifier[6]= nn.Linear(in_features=4096, out_features=2)
```

A binary cross-entropy loss is created using the `BCEWithLogitsLoss()` function from the `torch.nn` module.

In the final preparations for model training, an Adam optimizer is created using the `Adam()` function from the `torch.optim module`, which contains all of PyTorch's available optimizers.

```
optimizer= optim.Adam(vgg16.parameters(), lr=1e-4)
```

ResNet-18 Implementation

The next model to implement is ResNet-18, which can also be initialized from the `torchvision.models` package. Like VGG-16, we begin by importing the ResNet-18 model along with pre-trained weights from ImageNet.

```
from torchvision.models import resnet18, ResNet18_Weights
resnet18= resnet18(weights=ResNet18_Weights.IMAGENET1K_V1)
```

To prepare the model for binary classification, we can update the final fully connected layer to a linear transformation layer with two output features. With ResNet, this layer is accessed using the `fc` attribute.

```
resnet18.fc= nn.Linear(in_features=512, out_features=2)
```

A Binary CrossEntropyLoss function is created using the `BCEWithLogitsLoss()` function from the `torch.nn` module, in exactly the same way as with VGG-18. In fact, it is only necessary to define the loss function once and use it for all three models.

```
criterion = nn.BCEWithLogitsLoss()
```

For consistency and fair comparison, an Adam optimizer with the same learning rate as VGG-16 is used for the ResNet-18.

```
optimizer= optim.Adam(resnet18.parameters(), lr=1e-4)
```

MobileNetV2 Implementation

The final model to implement in our study is MobileNetV2. Once again, it should be imported and initialized with pre-trained weights from the ImageNet database.

```
from torchvision.models import mobilenet_v2, MobileNet_V2_Weights
mobilenet_v2= mobilenet_v2(weights=MobileNet_V2_Weights.IMAGENET1K_V1)
```

Binary classification preparation is achieved exactly as with VGG-16 using the classifier module. In this case the final fully connected layer sits is referenced with index 1.

```
mobilenet_v2.classifier[1]= nn.Linear(in_features=1280, out_features=2)
```

The loss function and optimizer are again the same as with the other models to ensure consistency and fairness.

```
criterion = nn.BCEWithLogitsLoss()
optimizer= optim.Adam(mobilenet_v2.parameters(), lr=1e-4)
```

Training the Models

We iterate through the dataset to train the models for a pre-determined number of epochs. Each epoch involves setting the model to training mode, which is set using the `train()` method. After training each model for 12 epochs, it was noted that there was no significant decrease in loss or increase in accuracy after 10 epochs. For this reason, each of the models was trained for 10 epochs to ensure fair comparison.

The images are then passed from the DataLoader to the GPU. After attempting to train the models with varying numbers of epochs, 10 was the number chosen, which still produced notable learning before approaching a plateau. Each epoch involves resetting the gradient `optimizer.zero_grad()`, passing the batch through the CNN, calculating the loss, and backpropagating using the `loss.backward()` method. Finally, the weights in the classification layer are adjusted with the `optimizer.step()` method.

We also keep track of a running loss to log the models' predictions over time. If the loss stagnates or begins to rise, we might consider tuning hyperparameters (like the learning rate) or stopping the training process early to avoid overfitting. As can be seen by the training output of VGG-16 in the following figure, the loss rate was significant enough for 10 epochs to warrant their execution. The `tqdm` library was used to monitor the batch progress visually, which made it easier to confirm that training was taking place and that plateaus were not being reached.

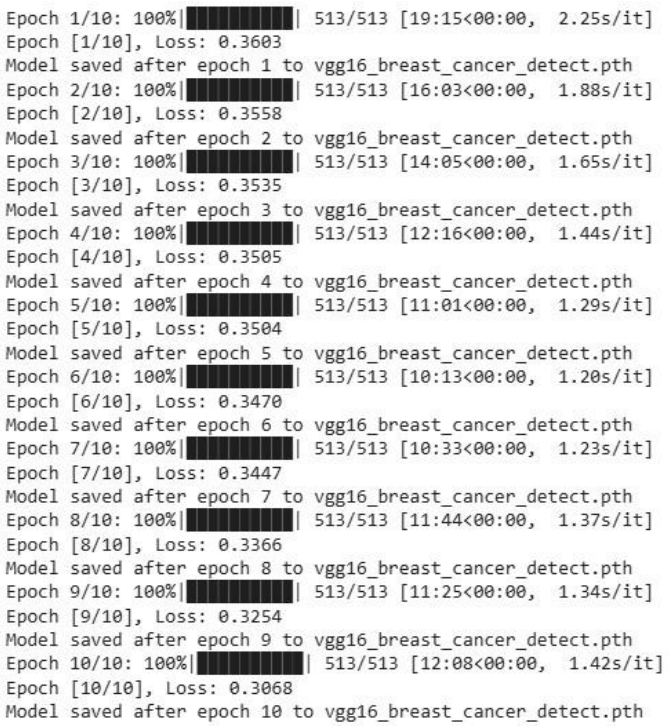

Figure 7: Initial training epochs VGG-16

Handling Class Imbalance

Following the first training phase, one of the main challenges when working with the VinDr-Mammo dataset presented itself. That is a significant class imbalance favoring benign cases over malignant cases. This is not an unknown problem when dealing with radiography images relating to cancer, since many of the images will have been scanned as a precaution and not because there were any certain signs of malignancy. The distribution of classes in the VinDr-Mammo training dataset is shown in the following figure.

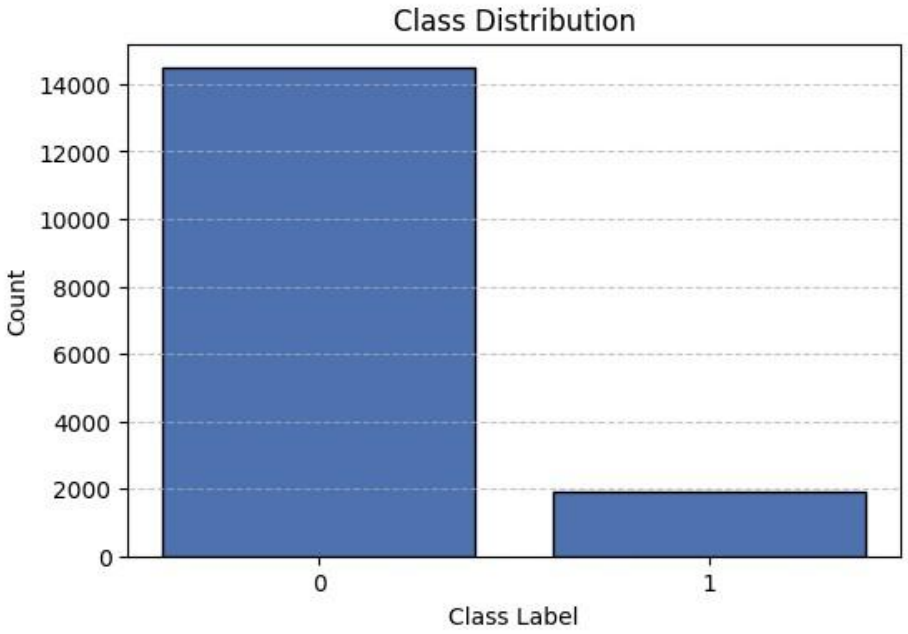

Figure 8: VinDr-Mammo Benign(0) vs. Malignant(1) class imbalance

The initial training phases for this study showed the detrimental impact that class imbalance can have on a model's predictions. Perhaps the worst part about this effect is that it will likely result in poor accuracy for underrepresented groups in the dataset. This was true for this study, which showed extremely accurate results when predicting benign cases but extremely inaccurate results when predicting malignancy. This is, of course, the worst-case scenario for a cancer detection model.

Multiple methods can address class imbalance, typically divided into two categories: undersampling and oversampling. Undersampling is the more straightforward technique, reducing well-represented classes to match the size of underrepresented classes. This was the first approach tested on the MobileNetV2 model for speed of training and computational efficiency. The following confusion matrix shows that some improvement was made, but the results are still poor.

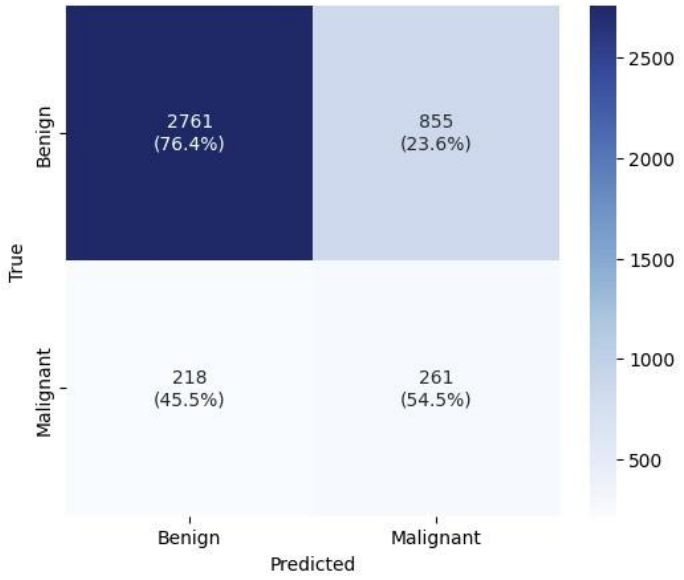

Figure 9: MobileNetV2 Confusion Matrix following undersampling.

Oversampling is more challenging to achieve, requiring replicating or synthesizing instances from the underrepresented class. A 2018 paper from Buda, M. et al. compared methods of tackling class imbalance and discovered that oversampling consistently produced greater accuracy, particularly in datasets with large imbalance ratios. Considering these findings, this study will address class imbalance in the VinDr-Mammo dataset with PyTorch's WeightedRandomSampler *(PyTorch, 2025).*

The WeightedRandomSampler accepts a tensor containing the class ratio, which is used to shift more of the loading probability to the underrepresented class. It is best practice to calculate the class weight ratio as in the following code snippet, although the tensor may still be hard-coded.

```
class_counts = torch.bincount(torch.tensor
      (train_dataset.labels, dtype=torch.long))
class_weights = 1.0 / class_counts.float()
sample_weights = class_weights[train_dataset.labels]

sampler = WeightedRandomSampler(
      weights=sample_weights,
      num_samples=len(sample_weights),
      replacement=True
      )
```

It should be noted that all of the models reacted positively to the WeightedRandomSampler. Given the severity of the class imbalance in VinDr-Mammo, an additional measure was implemented in the form of a weighted binary cross-entropy loss function. ResNet18 reacted well to the additional bias towards the underrepresented malignancy class, but MobileNetV2 and VGG-16 both shifted to extreme malignancy

bias, to the detriment of overall accuracy. This is represented in the following confusion matrix for VGG-16 with both measures applied.

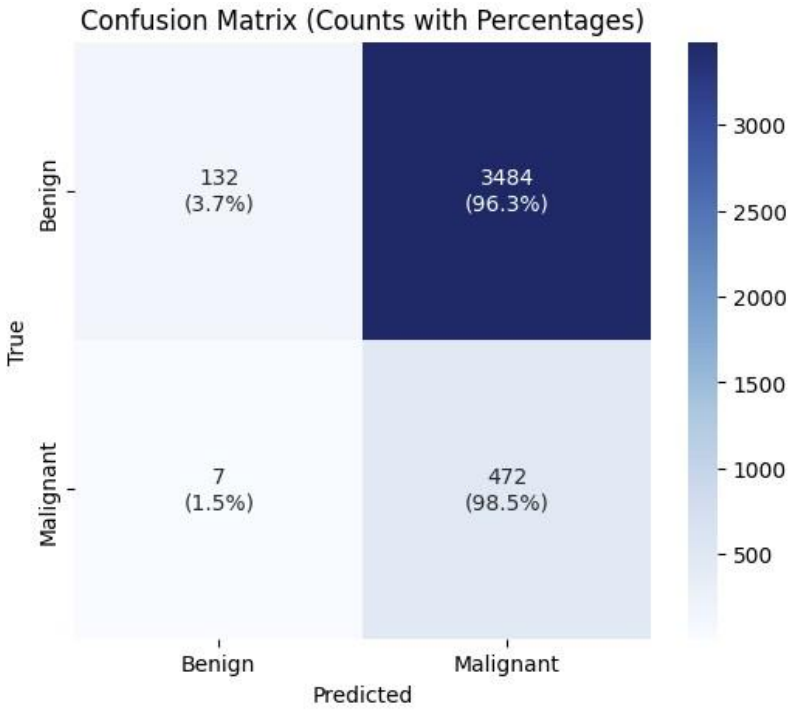

Figure 10: Class imbalance overcompensation resulting in extreme malignancy bias.

Considering this overcompensation, weighting was removed from VGG-16 and MobileNetV2. The weighted loss function was preserved for ResNet18, the model that had achieved the greatest overall accuracy so far.

The model's tendency for overfitting influenced the number of epochs decided upon for the next stage of training. Overfitting was evident with each of the models, whose training loss decreased steadily as expected, but validation loss began to diverge and fluctuate following 5 epochs. At this point, training has been optimized for the NVIDIA A100 GPU with large batch sizes and larger learning rates.

An initial learning rate of 0.0001 was too slow for the limited computational resources and did not leverage the A100 to its full potential. A learning rate of 0.0003 was eventually settled on for its balance of learning stability and computational efficiency.

Model Performance Evaluation

With the models trained and saved, it is time to investigate and compare their performance in binary classification. This investigation will aim to compare the accuracy and reliability of the models, paying particular attention to their intended use in the medical industry.

Before we begin evaluating and comparing the models, we must first decide what classifies a model as being ‘more reliable’ or ‘more precise’ for this application. As with most machine learning projects, the first metric for evaluation in this study is accuracy. Accuracy is simply a measure of the correct predictions made by the model in classifying test images as either malignant or benign. This can be used as a rough gauge of model performance when selecting and fine-tuning models.

However, it is critical to undergo a more detailed analysis of the models’ performance, particularly for models that aim to make medical diagnostic predictions. When selecting metrics for performance comparison, our primary concern is to minimize the risk of malignant anomalies going unnoticed. This study will utilize a number of metrics to achieve a thorough and fair performance evaluation for each model. This will ultimately ensure a fair comparison and final conclusion.

Confusion Matrix

Generating a confusion matrix for each model will allow us to break down the models' performance on a more granular level. Notably, this will allow us to identify ‘false negatives’ and avoid unidentified malignancy. False positives, true positives, and true negatives are also represented visually. Since our problem is one of binary classification, the matrix will be an uncomplicated 2x2 grid representing the results for the predicted labels (Malignant/Benign). After addressing the class imbalance and making minor optimizations, the following matrices show the results for all three models. The accuracy at this stage is not even close to the accuracy that might be expected of a medical diagnostic tool, but we can at least compare the pre-trained CNNs for their performance, having trained only their classification layer on the VinDr-Mammo dataset.

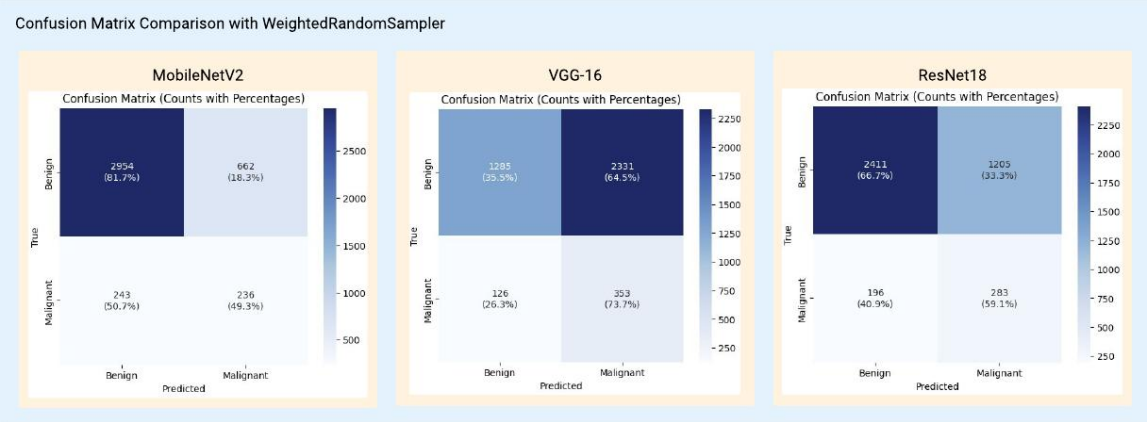

Figure 11: Confusion matrix comparison with WeightedRandomSampler

Precision

This metric refers to the percentage of all positive (malignant) predictions that were actually positive. Although this metric is useful for determining a model's probability of predicting false positives, the model's precision results should not significantly influence this study's comparison phase. That is because it is not as big of an issue for the model to predict malignancy when it doesn't exist as it is to miss it.

Recall

Recall, also referred to as the true positive rate, measures how many of the actual positive cases in the test data were correctly identified as positive. This metric is crucial in our analysis, as it's crucial to identify true positives correctly. However, as seen with the overcompensation that occurred when both the weighted loss function and WeightedRandomSampler were applied to VGG-16, recall alone is insufficient to evaluate the model's overall performance.

F1-Score

While precision and recall offer valuable insights individually, the F1-score incorporates both metrics for a balanced performance indicator regarding false positives and false negatives. Although false negatives are of particular concern in a medical context, both metrics hold some relevance to this study, making the F1-score a valuable metric.

Model	Weights	Accuracy	Precision	Recall	F1-score
VGG16	ImageNet	39.44 %	0.1264	0.7098	0.2146
ResNet18	ImageNet	65.89 %	0.1866	0.5699	0.2812
MobileNetV2	ImageNet	77.66 %	0.2632	0.4781	0.3395

Figure 12: Table of model performance metrics

Although MobileNetV2 achieves the highest accuracy and F1-score, it exhibits the lowest recall, which is critical for a cancer detection model. Given this, it seems more logical to proceed with ResNet. However, its performance remains relatively poor, indicating there is still significant room for optimization. The next step would be to unfreeze some of ResNet18's final layers and apply transfer learning to the VinDr-Mammo dataset. Due to the large size and high computational cost of both the models and the dataset, training has been extremely resource-intensive. The cost of reaching this stage has exceeded €68.26 for GitHub Colab Pro access. Breast BI-RADS ratings are notoriously challenging to classify, as highlighted in *'Artificial intelligence and convolutional neural networks assessing mammographic images: A narrative literature review' (Wong et al., 2020).* At this stage, the findings of this study suggest that the pre-trained CNNs may not yet be sufficient for this task.

Some studies have shown relatively high accuracy in predicting BIRADs with architectures such as VGG and ResNet, but with greater computational resources for optimized fine-tuning and transfer learning. To simulate the fine-tuning of the ResNet model with limited resources a ResNet model trained pre-trained on medical images from the Rad Image Net database could be used *(Mei et al., 2022).* A ResNet50 backbone containing RadImageNet was taken from the following github repository: https://github.com/BMEII-AI/RadImageNet/tree/main

As with ResNet18 the classification layer was updated and trained against the VinDr-Mammo dataset. In fact ResNet50 did not come with a classification layer given it's intended use as a transfer learning backbone. Unlike ResNet18, the model reacted badly to the application of both a weighted loss function and WeightedRandomSampler. Training ResNet50 for a second time with just the WeightedRandomSampler and a standard binary cross-entropy loss function proved to suit the model better as can be seen in the following confusion matrices and updated table of performance metrics.

Model	Weights	Accuracy	Precision	Recall	F1-score
VGG16	ImageNet	39.44 %	0.1264	0.7098	0.2146
ResNet18	ImageNet	65.89 %	0.1866	0.5699	0.2812
MobileNetV2	ImageNet	77.66 %	0.2632	0.4781	0.3395
ResNet50	RadImageNet	68.72%	0.1722	0.4551	0.2499

Figure 13: Table of model performance metrics with ResNet50

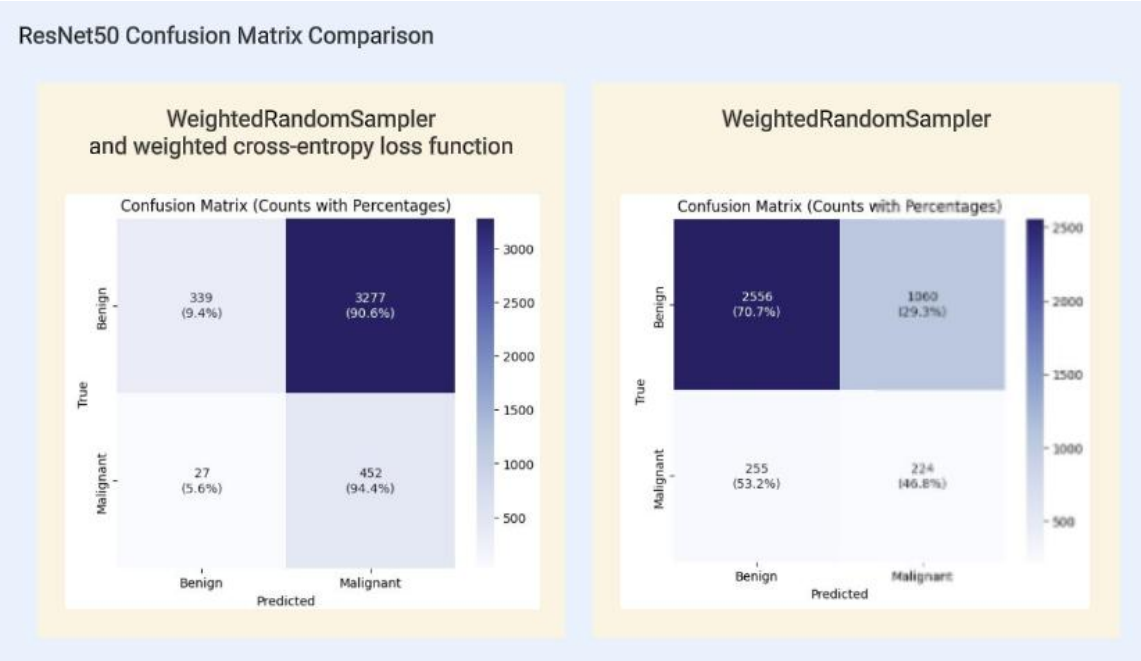

Figure 14: ResNet confusion matrix comparison

Comparison of results

Although none of the pre-trained models have achieved any great accuracy in the binary classification of mammograms, there are some valuable insights to be gained from comparing their results. Most notably, the MobileNetV2 architecture has achieved the greatest overall accuracy despite being the most lightweight model. This is, however, offset by the model recall result, which was the lowest measured. VGG-16 had the highest recall, meaning the highest correctly predicted malignant cases, which was identified as a crucial factor for model selection in this study.

ResNet18 was identified as the most balanced performer, prompting the study to continue with a ResNet architecture. Since none of the models achieved great results with the pre-trained ImageNet weights, a ResNet50 model was loaded with weights from pre-training on a medical image database. ResNet50 did not achieve any improvements, scoring less than the original ResNet18 model under each metric.

Ethical and Bias Assessment

As with any Artificial Intelligence project in 2025, ethics and bias should be considered from its conception to completion. This section will interrogate the steps taken, commenting on strong points and, more importantly, identifying areas for improvement. It is intended that conclusions and suggestions made in this section can be applied to future iterations of the project or projects with similar characteristics, following the framework and guidelines set by the Assessment List for Trustworthy Artificial Intelligence (ATLAI). ATLAI is a framework created by the High-Level Expert Group on Artificial Intelligence, which was first opened by the European Commission in 2018. The group's intention with ATLAI is to provide organizations with a basis for evaluating the trustworthiness and risks associated with an AI system *(European Commission High-Level Expert Group on Artificial Intelligence, 2020)*. Considering the medical context of this study, it is imperative to be thorough in this ethical assessment. While the predictions made in this study are intended for exploratory purposes only, this study aims to find a direction for future research into using machine learning for early breast cancer detection. Ultimately, the goal will be to create a model that radiologists may use to streamline their mammogram screening process. Such a significant disruption to the conventional screening process must prioritize all patients' fair and equal treatment while ensuring trust and confidence in the prediction model.

Human Agency and Oversight

ATLAI guidelines for trustworthy AI state that AI systems should support human autonomy and decision-making, contributing to a democratic society *(EC HLEG on AI, 2020)*. It is essential that models aiming to detect signs of malignancy in mammograms do not replace the autonomy of human judgement in the radiographer's decision-making process. The models are intended only to identify malignancy cases earlier, not as a final decision-making tool. Furthermore, predictions made by the models are not intended to support the decision-making process of the radiographer. This is due to the lack of explainability with convolutional neural networks (CNNs). Although there are models with far greater explainability, their performance in medical image analysis does not come close to that of CNNs.

This study suggests that CNNs are the best choice for early detection, but only with the proper systems in place to ensure that practitioners may never rely on them for decision-making. Such a system should ensure complete transparency of the decision-making process for patients. Full transparency will reduce confusion about the source of decisions for patients. The assurance of such a system is certainly a challenge, but the rewards are too great to ignore.

Technical Robustness and Safety

Dependability of an AI system, in terms of technical robustness and safety, forms another pillar of the ATLAI guidelines for trustworthy AI. As with all ethical assessments made in this section, the topic of dependability in reliable, robust solutions is particularly relevant in a medical context. A robust AI system is one that is secure from external cyber threats, ensures accuracy that won't lead to damaging consequences, and provides general safety relating to the system's use *(EC HLEG on AI, 2020).*
It is important to note that the models developed for this study are exploratory and, as such, would fail an assessment into each of these areas.

This study concluded that sufficient accuracy cannot be achieved from pre-trained backbone models through transfer learning on a small scale. Regarding security, the trained models are currently protected by the security measures put in place by Google Cloud Storage, where they reside. Ultimately, models should remain open-source to ensure transparency and human oversight. Still, the applications that use them will require strict security measures to ensure patient confidentiality and prevent malicious interference. Regarding general safety, a system surrounding the decision-making process must be put in place before early detection using ML is incorporated into mammogram screening procedures.

Privacy and Data Governance

The data used to train the CNNs in this study has been stripped of all identifiable information relating to the examined patients. This is a non-negotiable characteristic of the data that should be used to train medical image analysis models. None of the work carried out in this study places additional risk of being included in the dataset in the first place. The dataset is publicly available for anyone who agrees to the terms and conditions outlined by VinDr-Mammo.

Further measures would be required to safeguard the patient's personal information in a production environment. A far greater risk occurs when the model is used to aid practitioners in the early detection of breast cancer because the results would likely be stored with a patient's sensitive medical file. It is, therefore, crucial that data sent to the model is fully non-identifiable, with personal information remaining inside the practitioner's local environment, which we must presume to be secure.

Transparency

Perhaps the most significant challenge that early breast cancer detection with convolutional neural networks presents is the lack of transparency and explainability of its learning and predictions. The following chart was taken from Yang, G., Ye, Q., and

Xia, J.'s 2021 paper, 'Unbox the black box for the medical explainable AI via multi-modal and multi-centre data fusion: A mini-review, two showcases and beyond'.

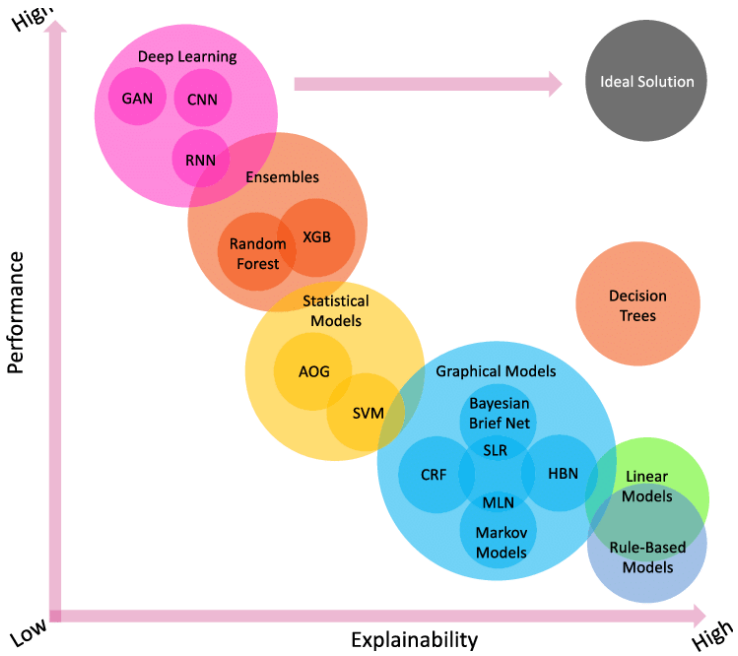

Figure 15: Model explainability chart (Yang, Ye & Xia, 2021)

This diagram visually confirms the claim that CNNs offer the greatest medical image analysis accuracy but the least explainability. The graph shows a linear correlation between explainability and performance, with the exception of decision trees, which offer explainability as high as linear and rule-based models, with performance that remarkably approaches that of the ensemble models, such as XGBoost. The accuracy does, however, fall short of that which CNNs can achieve. That being said, there is an argument to be made that Decision Trees may find greater application in medical image analysis with limited resources. This would preserve full autonomy for practitioners and patients, particularly in disadvantaged regions. Theoretically, a robust solution with full explainability could be used for early detection and to support decision-making.

Diversity, Non-discrimination and Fairness

Trustworthy AI systems are those that hold no bias toward particular groups of people. The data used in this study were gathered from a number of hospitals in Vietnam, meaning that mammograms were likely from patients of the major ethnic groups in Vietnam. The primary ethnic group in Vietnam is the Kinh people, who account for 87% of the population *(Empire of the Socialist Republic of Vietnam in the United States, 2025)*. By training breast cancer detection models using only the VinDr-Mammo

dataset, there is potential bias toward the ethnic groups included in the dataset.

Future iterations of the models in this study should attempt to diversify the training data by including mammogram sources from varying locations and with diverse ethnic backgrounds. This is an important measure to ensure fair prediction for all patients.

In terms of accessibility and unfair bias, the explorations in this study aim to contribute to identifying a direction for medical image analysis that is accessible to all. This involves considering techniques that can be implemented with limited resources, ensuring that the foundational structure of impactful technologies, such as breast cancer detection, remains democratic from research to production.

Societal and Environmental Well-being

Convolutional Neural Networks are notoriously computationally expensive to train. Appropriate selection of computational resources and infrastructure can majorly influence the environmental and societal costs of training. This study utilized Google Cloud as an infrastructure solution that is accessible to all, ensuring that model development is not reserved for those with access to expensive computer systems.

The development of machine-learning medical diagnostic systems poses an undeniable threat to the work of medical practitioners in a wide range of fields, such as radiographers and mammography specialists. This study emphasizes that the intention of such systems should be to aid in early detection, and potentially, for models with greater explainability, to support the decisions of human practitioners. Preserving the final decision for a qualified practitioner not only ensures long-term diagnostic reliability but also protects the work and well-being of practitioners in the field.

A significant benefit of using Google Cloud is the ability to measure the carbon footprint of services used and produce real-time reporting on carbon insights using Google Cloud Carbon Footprint Exports. Measurement is the first step to management, and doing so will allow researchers to understand the environmental impact of models used and any fine-tuning steps or optimizations applied. As a cloud service provider, Google has claimed carbon neutrality since 2007, with investment in carbon offset projects such as landfill gas recovery, where methane is converted to electrical energy *(Google Inc., 2011).*

Accountability

Implementing a version control system such as git is essential to ensuring accountability in this study. Adding git to the project will ensure that a reliable version history tree is maintained, with individual contributors associated with each change in revision or

commit. Without transparency relating to software changes, there can be no accountability for decisions made in logic changes or model fine-tuning. Initiating a strict version control process from the initial commit will ensure that internal and external auditors can audit the development history with confidence in its authenticity and accuracy.

Moving all model development and training to the cloud further optimizes auditing. Auditors may be added as principals of type 'Audit Manager Admin' under the Identity and Access Management settings for cloud services. This grants appropriate access to the service's data and additional auditing resources.

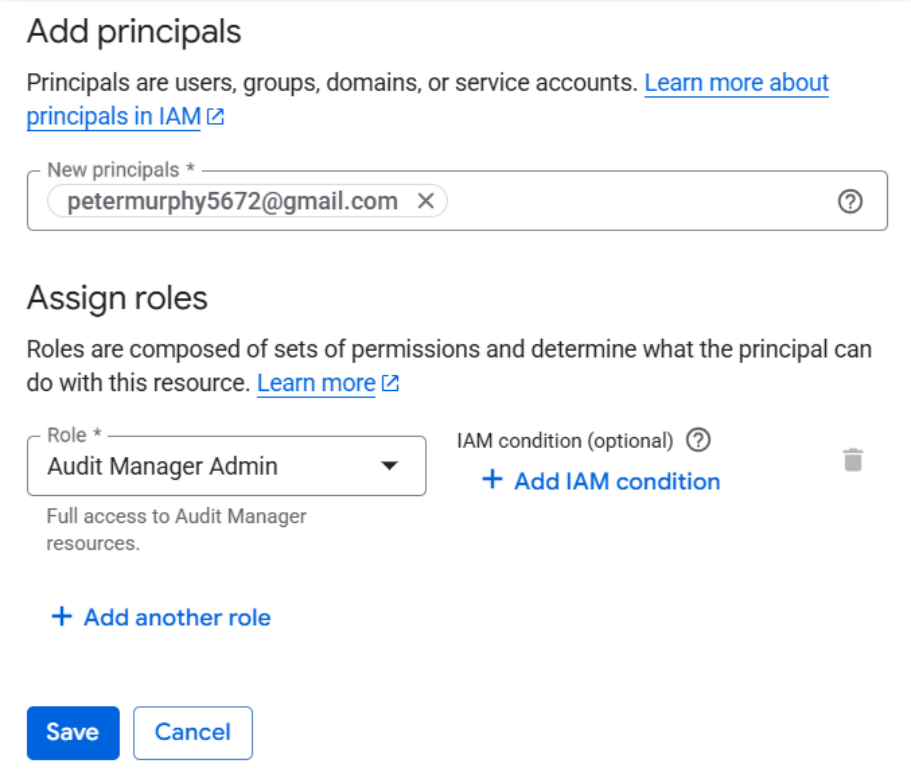

Figure 16: Add auditing principal to Google Cloud Service
(Screenshot from Google Cloud Console)

Regular auditing should represent a step in a comprehensive risk management strategy. This ethical assessment has identified risks such as prediction bias due to a lack of dataset diversity, practitioner reliance on predictions from models lacking explainability, and malicious interference from cyber threats. A comprehensive risk management strategy would consist of regular auditing, monitoring model performance and security vulnerabilities, and promoting the explainability of AI models.

This ethical assessment has highlighted the many positive impacts that machine learning can have in medical image analysis. However, these benefits come with challenges and risks that need to be carefully managed to maximize the technology's overall benefit. The steps taken to identify ethical concerns and risks to public welfare are essential for ensuring a positive and lasting relationship between the fields of machine learning and medicine.

Production

This section will discuss training the selected CNNs in a cloud environment and migrating the trained CNN models to a production environment on Google Cloud. Given the substantial size of the VinDr-Mammo dataset, the training and deployment of the models require the use of advanced cloud infrastructure. Google's Colaboratory Pro platform offers access to some of the most powerful GPUs available, making it a perfect platform for efficient model training. The large-scale VinDr-Mammo dataset requires a scalable and efficient storage solution to accompany the enhanced computing resources. Google Cloud Storage buckets will provide the necessary storage resources for the dataset, ensuring accessibility and reliability throughout the training and deployment phases.

Training Runtime Device

Due to the large number of images involved in training the CNN models for this study, the estimated training time using a standard Intel i7 CPU was 110 hours for VGG-16. With a Google Colab Pro account, it is possible to select from various devices that are far better suited to training ML and DL models. To achieve optimum efficiency, the NVIDIA A100 Tensor Core GPU was selected.

```
+---------------------------------------------------------------------------------------+
| NVIDIA-SMI 550.54.15              Driver Version: 550.54.15      CUDA Version: 12.4     |
|-----------------------------------------+------------------------+----------------------+
| GPU  Name                 Persistence-M | Bus-Id          Disp.A | Volatile Uncorr. ECC |
| Fan  Temp   Perf          Pwr:Usage/Cap |           Memory-Usage | GPU-Util  Compute M. |
|                                         |                        |               MIG M. |
|=========================================+========================+======================|
|   0  NVIDIA A100-SXM4-40GB          Off |   00000000:00:04.0 Off |                    0 |
| N/A   31C    P0             51W /  400W |   10541MiB /  40960MiB |      0%      Default |
|                                         |                        |             Disabled |
+-----------------------------------------+------------------------+----------------------+

+---------------------------------------------------------------------------------------+
| Processes:                                                                            |
|  GPU   GI   CI        PID   Type   Process name                            GPU Memory |
|        ID   ID                                                             Usage      |
|=======================================================================================|
+---------------------------------------------------------------------------------------+
```

Figure 17: NVIDIA A100 Tensor Core GPU Specifications

Using the `torch.device("cuda")` command, we indicate to PyTorch that tensors should be routed to the GPU for CUDA operations. CUDA (Compute Unified Device Architecture) is a framework developed by NVIDIA to accelerate the efficiency of GPUs with advanced parallel computing techniques, which have been optimized for machine learning and deep learning applications. With these adjustments, the estimated training time for the VGG-16 model decreased from 110

hours to 2 hours.

Training Data Storage

Now that a runtime environment has been defined in Google Colaboratory with access to the appropriate hardware, we need to ensure that we can effectively access our training and test data. Since it is not possible to efficiently access data saved to a local machine from a Google Colab notebook, the data necessary for training the model should be stored in a Google Cloud Storage bucket. Uploading larger folders can be tricky when using GCS's online user interface since progress isn't always displayed. For this reason, it is easier to download the Google Cloud SDK and transfer the folders from the command line:

```
gsutil -m cp -r /path/to/local/folder gs://your-bucket-name/
```

Authenticating access between Colab and the storage bucket is easily achieved by generating an access key for your Google Cloud service user and assigning it to an environment variable:

```
os.environ["GOOGLE_APPLICATION_CREDENTIALS"] = "/path/to/key.json"
```

When working with a GCS bucket, we must first initialize a client using storage.Client() in the training runtime before attempting to access data from the bucket. The bucket can then be accessed using the `client.bucket()` method.

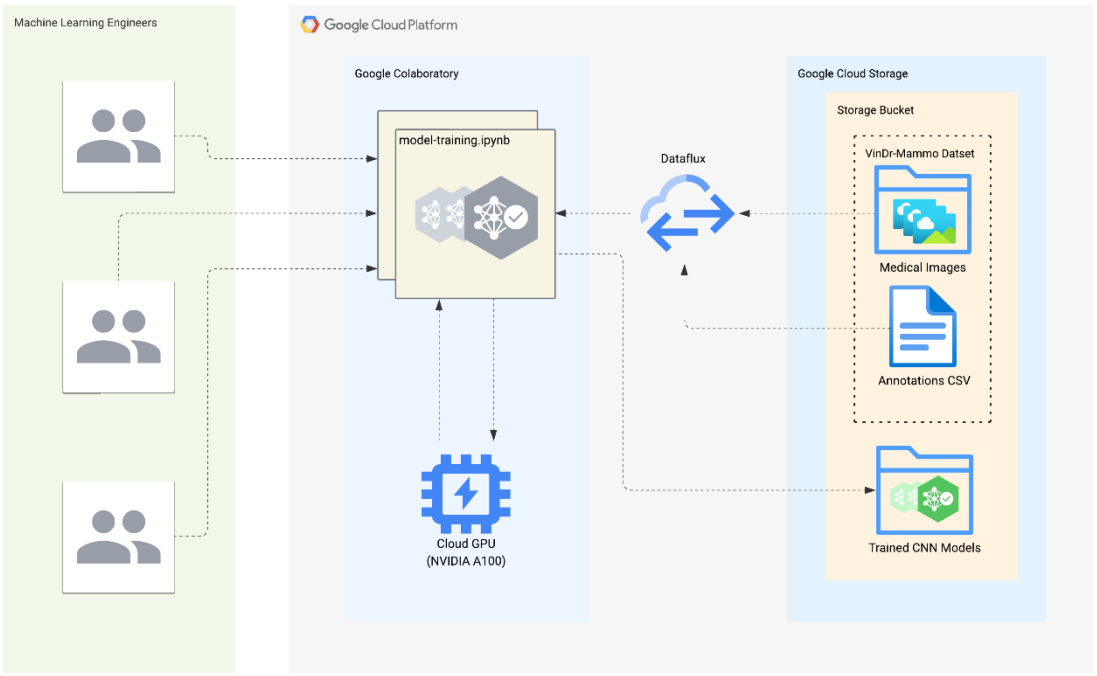

Figure 18: Cloud Architecture for Model Training using Google Cloud Platform

Model Deployment

After the models have been trained, evaluated, and compared, the most appropriate model for our breast cancer detection application is ready for deployment to a production environment. The model will be accessed through a REST API built using a simple and scalable framework, which can handle requests and return predictions. The API will be hosted on a reliable cloud-based service to ensure ease of access, security, and scalability.

Building the REST API

Flask is a lightweight Python web application framework commonly used for deploying machine learning models in production due to its minimalistic and flexible design. It makes creating a REST API that interacts with a trained model easy and allows other systems or users to make predictions. Flask is preferred over larger frameworks such as Django because it demands fewer resources and is easier to set up for simple ‘proof-of-concept’ applications like this one.

At its core, the application will include an endpoint that accepts mammogram images as input, classifies the images using the trained model, and returns the

prediction result (benign or malignant). This will allow external medical diagnostic systems, or users, to send an image via an HTTP POST request and receive a response with the model's prediction. In the future, additional endpoint parameters could be added to adjust the model's parameters for optimized training and predictions.

The first step to creating the breast cancer prediction REST API is to install Flask using `pip install Flask`. The `main.py` file will contain the logic for handling requests and serving the trained model predictions.

The basic structure of the breast cancer prediction API begins with initializing an instance of the `Flask` class. Initializing the instance requires a first parameter of `__name__`, which is a Flask requirement and defines the application's module. The primary application route can then be established using the `@app.rotue("/")` decorator. This route will serve predictions based on the VinDr-Mammo trained model *(Flask Documentation, n.d.).*

Deploying the REST API

Regarding a cloud solution that supports elastic scaling based on request numbers and model instances that may need to be run, Google Cloud Run offers a suitable solution for this application. Google Cloud Run is a service that operates under a 'scale to zero' policy, meaning that server instances are removed when there are no incoming requests. This ensures that computational power isn't wasted and that costs are reduced.

The first step in application deployment is to set up a project using the Google Cloud Console. Considering that we have no particular need for isolation in this case, it makes sense to deploy the application to the same project that our training data was stored in. Utilizing the same project for this simple 'proof-of-concept' will reduce complexity and remove the need for duplicated permissions and billing configurations. Another benefit is that environment variables are accessible across services inside the same project.

Once inside the project in the Console CLI, the root directory and 'main' file for the REST application can be created.

```
mkdir breast-cancer-detection-api
cd breast-cancer-detection-api
touch main.py
```

The completed Flask application is now ready for containerization and deployment to Google Cloud Run.

Containerizing and Deploying the Application

Google Cloud Run requires that the application be first containerized. Containerization is ordinarily achieved using Docker, allowing us to specify an environment and dependencies suited to the application. Google Cloud's CLI provides a `source` option with its `deploy` tool that automatically builds the container image from our source code as part of the deployment process. The command can be run with the path to our application's root folder as a final parameter, or run from within the folder using '.' To specify the current directory *(Google Cloud, 2025)*.

```
gcloud run deploy --source path/to/flask/app
```

After specifying a name and a region from the CLI, a service URL will be returned, through which the application can be viewed and the Flask endpoints accessed.

Conclusion

This study evaluated the performance of pre-trained CNNs for classifying breast cancer using the VinDr-Mammo dataset. While the models did not achieve high accuracy, some key insights were gained.

The class imbalance in the dataset significantly affected performance, with models tending to predict benign cases more often. However, a combination of techniques, like data augmentation, reduced the impact of the imbalance on key performance metrics.

Resource limitations also impacted the training process. Although Google's Colab made available powerful GPUs, training deep CNNs on large datasets is slow and costly. Future work should focus on more efficient methods to improve model performance within these constraints. Working towards a method of medical image analysis with limited resources will improve accessibility.

Another challenge is posed in addressing the lack of explainability in the models' decision-making process. While CNNs like MobileNetV2 and particularly ResNet-18 showed some promise, their lack of transparency is an issue for medical applications. Models with better explainability, like decision trees, should also be explored.

To conclude, although the models did not perform as expected, the study highlights some important insights for future research.

References

1. World Health Organization (2024). *Breast Cancer*. Available at: https://www.who.int/news-room/fact-sheets/detail/breast-cancer [Accessed 11 November 2024].

2. Asri, H., Mousannif, H., Al Moatassime, H. & Noel, T. (2016). 'Using Machine Learning Algorithms for Breast Cancer Risk Prediction and Diagnosis'. *Procedia Computer Science*, 83, pp. 1064–1069. https://doi.org/10.1016/j.procs.2016.04.224.

3. Pham, H. H., Nguyen Trung, H., & Nguyen, H. Q. (2022). VinDr-Mammo: A large-scale benchmark dataset for computer-aided detection and diagnosis in full-field digital mammography (version 1.0.0). PhysioNet. https://doi.org/10.13026/br2v-7517

4. Murel, J. and Kavlakoglu, E. (2024). 'What is transfer learning?'. *IBM*. Available at: https://www.ibm.com/think/topics/transfer-learning [Accessed 15 April 2025].

5. Nguyen, H.T., Nguyen, H.Q., Pham, H.H., Lam, K., Le, L.T., Dao, M., & Vu, V. (2022). 'VinDr-Mammo: A large-scale benchmark dataset for computer-aided diagnosis in full-field digital mammography', medRxiv. Available at: https://doi.org/10.1101/2022.03.07.22272009 [Accessed 12 December 2024].

6. Carrilero-Mardones, M., Parras-Jurado, M., Nogales, A. _et al._ Deep Learning for Describing Breast Ultrasound Images with BI-RADS Terms. J Digit Imaging. Inform. med. 37, 2940–2954 (2024). https://doi.org/10.1007/s10278-024-01155-1

7. Asri, H., Mousannif, H., Al Moatassime, H., & Noel, T. (2016). Using Machine Learning Algorithms for Breast Cancer Risk Prediction and Diagnosis. Procedia Computer Science, 83, 1064-1069. https://doi.org/10.1016/j.procs.2016.04.224

8. DICOM (2024). Homepage. Available at: https://www.dicomstandard.org/ [Accessed 7 December 2024].

9. BreastCancer.org (2024). The Breast Imaging Reporting and Data System (BI-RADS). Available at: https://www.breastcancer.org/screening-testing/mammograms/bi-radsresults [Accessed 11 November 2024]

10. PyDicom (2023) PyDicom: Python package for DICOM medical file reading and writing. Available at: https://pydicom.github.io/ [Accessed 27 April 2025]

11. Simonyan, K. and Zisserman, A., 2015. Very deep convolutional networks for large-scale image recognition. *arXiv preprint* arXiv:1409.1556v6. Available at: https://doi.org/10.48550/arXiv.1409.1556 [Accessed 17 April 2025].

12. Sugata, T & Yang, C. (2017). Leaf App: Leaf recognition with deep convolutional neural networks. IOP Conference Series: Materials Science and Engineering. 273. 012004. 10.1088/1757-899X/273/1/012004.

13. He, K., Zhang, X., Ren, S. & Sun, J., 2015. Deep Residual Learning for Image Recognition. *arXiv*. Available at: https://doi.org/10.48550/arXiv.1512.03385 [Accessed 17 April 2025].

14. Russakovsky, O., Deng, J., Su, H. *et al.* ImageNet Large Scale Visual Recognition Challenge. *Int J Comput Vis* **115**, 211–252 (2015). https://doi.org/10.1007/s11263-015-0816-y

15. Howard, A.G., Zhu, M., Chen, B., Kalenichenko, D., Wang, W., Weyand, T., Andreetto, M. and Adam, H., 2017. MobileNets: Efficient Convolutional Neural Networks for Mobile Vision Applications. *arXiv preprint arXiv:1704.04861* [online]. Available at: https://arxiv.org/abs/1704.04861 [Accessed 17 April 2025]

16. Akay, M., Du, Y., Sershen, C., Wu, M., Chen, T., Assassi, S., Mohan, C. & Akay, Y. (2021). 'Deep learning classification of systemic sclerosis skin using the MobileNetV2 model'. *IEEE Open Journal of Engineering in Medicine and Biology*, pp. 1-1. https://doi.org/10.1109/OJEMB.2021.3066097.

17. Stanford Vision Lab, Stanford University, and Princeton University, 2020. ImageNet. [online] Available at: https://imagenet.org [Accessed 17 April 2025].

18. Buda, M., Maki, A. & Mazurowski, M.A. (2018). 'A systematic study of the

class imbalance problem in convolutional neural networks'. *Neural Networks*, 106, pp. 249–259. https://doi.org/10.1016/j.neunet.2018.07.011 [Accessed 24 April 2025].

19. PyTorch (2025). *torch.utils.data.WeightedRandomSampler*. Available at: https://pytorch.org/docs/stable/data.html#torch.utils.data.WeightedRandomSampler [Accessed 24 April 2025].

20. Wong, D. J., Gandomkar, Z., Wu, W. J., Zhang, G., Gao, W., He, X., Wang, Y., & Reed, W. (2020). 'Artificial intelligence and convolutional neural networks assessing mammographic images: A narrative literature review'. *Journal of Medical Radiation Sciences*, 67(3), pp. 134–142. https://doi.org/10.1002/jmrs.385.

21. Mei, X., Liu, Z., Robson, P. M., Marinelli, B., Huang, M., Doshi, A., Jacobi, A., Cao, C., Link, K. E., Yang, T., Wang, Y., Greenspan, H., Deyer, T., Fayad, Z. A. & Yang, Y. (2022). 'RadImageNet: An open radiologic deep learning research dataset for effective transfer learning'. *Radiology: Artificial Intelligence*, 4(5), e210315. https://doi.org/10.1148/ryai.210315 [Accessed 27 April 2025].

22. European Commission High-Level Expert Group on Artificial Intelligence (2020). *Assessment List for Trustworthy Artificial Intelligence (ALTAI)*. Available at: https://ec.europa.eu/digital-strategy/our-policies/ethics-guidelines-trustworthy-ai_en [Accessed 27 April 2025].

23. Yang, G., Ye, Q. & Xia, J. (2021). 'Unbox the black-box for the medical explainable AI via multi-modal and multi-centre data fusion: A mini-review, two showcases and beyond'. *Information Fusion*, 77, pp. 1-15. https://doi.org/10.1016/j.inffus.2021.07.016.

24. Embassy of the Socialist Republic of Vietnam in the United States (n.d.). *Ethnic Groups*. Available at: https://vietnamembassy-usa.org/culture/ethnic-groups [Accessed 27 April 2025].

25. Google Inc. (2011). *White Paper: Google's Carbon Offsets*. Available at: https://www.google.com/green [Accessed 27 April 2025].

26. Flask Documentation. (n.d.). *Quickstart*. Available at: https://flask.palletsprojects.com/en/stable/quickstart/ (Accessed: 21 April 2025).

27. Google Cloud, 2025. *Quickstart: Deploy a Python service to Cloud Run*. Available at: https://cloud.google.com/run/docs/quickstarts/build-and-deploy/deploy-python-service [Accessed 23 April 2025].