Breast Cancer Classification

A Project Submitted

in partial Fulfilment of the requirements

for the Degree of

Bachelor of Technology

in

Computer Science and Engineering

As a part of the core elective course

"Deep Learning"

By

Chittepu Rohith Reddy - 200378

Maram Anvitha – 200399

Under the supervision of

Dr. Soharab Hossain

BML MUNJAL UNIVERSITY™

SCHOOL OF ENGINEERING AND TECHNOLOGY

BML MUNJAL UNIVERSITY GURGOAN

April 23, 2023

Declaration by the Candidates

We hereby declare that the project entitled "**Breast Cancer Classification**"; has been carried out for fulfilling the partial requirements for completion of the core-elective course on Deep Learning offered at the 6th Semester of the Bachelor of Technology (B.Tech) program in the Department of Computer Science and Engineering during AY-2022-23 (even semester). This experimental work has been carried out by us and submitted to the course instructor Dr. Soharab Hossain Shaikh. Due acknowledgments have been made in the text of the project to all other materials used. This project has been prepared in full compliance with the requirements and constraints of the prescribed curriculum.

Chittepu Rohith Reddy

Maram Anvitha

Place: BML Munjal University

Date: 23 April, 2023

Contents

1. Introduction

Breast cancer is a prevalent and significant health concern affecting women worldwide. As the precision medicine movement gets underway, automated breast cancer using histopathological images is important for clinical diagnosis and prognosis. Breast cancer is the most prevalent disease in women worldwide, with significant death and morbidity rates, according to the World Health Organization's (WHO) World Cancer Report. Timely detection and accurate diagnosis play a crucial role in improving patient outcomes and saving lives. In recent years, deep learning, a subset of machine learning, has emerged as a promising approach in breast cancer research, with the potential to enhance the accuracy and efficiency of breast cancer diagnosis.

Deep learning, including convolutional neural networks (CNNs), a subtype of deep learning, has shown great promise in breast cancer research. CNNs are specifically designed for image analysis tasks and have been extensively used for breast cancer detection, classification, risk prediction, and prognosis. In recent years, CNNs have been utilized in breast cancer research to analyze mammograms, magnetic resonance imaging (MRI), ultrasound, and histopathology images, among others. These networks can automatically learn and extract relevant features from medical images without relying on handcrafted features, making them highly effective in identifying subtle patterns and abnormalities associated with breast cancer.

This report presents a deep learning-based approach for breast cancer classification using a convolutional neural network (CNN). The methodology includes data pre-processing, data augmentation, data splitting, and the architectures of CNN and Transfer Learning model with VGG16, ResNet50V2 base. The results demonstrate the effectiveness of the proposed approach in accurately classifying breast cancer images, with an accuracy of 88%. The findings of this study have significant implications for the development of automated breast cancer diagnosis systems that can assist healthcare professionals in making accurate and timely diagnosis. The experimental results reported based on the performance metrics of Average Precision, Average Recall, and Average F1-Score, it appears that the CNN model achieved the highest accuracy among the three models tested, with an average precision of 88.63%, an average recall of 88.50%, and an average F1-score of 88.48%. The VGG16 model had slightly lower scores, with an average precision of 86.59%, an average recall of 86.57%, and an average F1-score of 86.57%. Similarly, the ResNet50V2 model also achieved similar scores, with an average precision of 86.66%, an average recall of 86.63%, and an average F1-score of 86.62%.

2. Problem Statement

In order to detect breast cancer early and provide effective therapy, it is essential to accurately classify the disease. However, even for skilled radiologists, it can be difficult to accurately interpret histopathology pictures. Breast cancer classification using convolutional neural networks (CNNs) has shown to be highly accurate in the field of medical picture analysis. It has also been demonstrated that the use of transfer learning, a method that makes use of pre-trained CNN models, enhances classification performance in medical imaging applications. In this project, we aim to investigate the use of CNNs and transfer learning for breast cancer classification using histopathology images, with the goal of improving the accuracy and efficiency of breast cancer diagnosis.

3. Literature review

Zhongyi Han et al. [1] proposed a breast cancer multi-classification method using a newly proposed deep learning model. The proposed structured deep learning model consists of two components: a CNN for image feature extraction which is designed to learn high-level features from the histopathological images and a SIN for structured information from the images. The main challenges are multiple classes due to broad variability and great difficulties in multi-classification methods of breast cancer differ from binary classes. The CSDCNN method for multi-classification of breast cancer from histopathology images involves three stages: training, validation, and testing. The model learns hierarchical features automatically from low-level to high-level representations during the training stage. Distance constraints are applied in the feature space to improve the model's discriminative capacity and multi-classification accuracy. The structured deep learning model performed remarkably well (average 93.2% accuracy) on a large set of data, showing the effectiveness of this method in offering an effective tool for breast cancer multi-classification in clinical settings.

This model uses two datasets for evaluating the performance of their proposed method. BreaKHis is a large-scale dataset of 7909 histopathological images of breast cancer from 82 anonymous patients, with four magnification factors and 700x460 pixels in RGB format. BreaKHis dataset augmented using data augmentation method to address imbalanced class problems. Eight classes of breast cancer histopathological images from BreaKHis dataset. There are challenging histopathological images due to the broad variability of high-resolution image appearances, high coherency of cancerous cells, and extensive inhomogeneity of color distribution. These histopathological images were all acquired at a magnification factor of 400.

Mohamed Sakr et al. [2] proposed a novel deep-learning model based on the transfer-learning (TL) technique for automatic detection and diagnosis of breast cancer using mammographic images.

The suggested model utilizes a fully connected neural network as a classifier and a pre-trained convolutional neural network (CNN) as a feature extractor. Due to the transfer-learning technique the model can use the knowledge gained from a large dataset to enhance performance on a smaller dataset. The experimental results reported in the paper suggest that the transfer learning (TL) approach using the VGG16 model is powerful for breast cancer (BC) diagnosis, achieving high accuracy, sensitivity, specificity, precision, F-score, and area under the curve (AUC) values are 98.96%, 97.83%, 99.13%, 97.35%, 97.66%, and 0.995, respectively for 80-20 method and 98.87%, 97.27%, 98.2%, 98.84%, 98.04%, and 0.993 for 10-fold cross-validation method.

The dataset used in the proposed model is Mammographic Image Analysis Society (MIAS) dataset. The MIAS dataset consists of 322 mammograms taken from 161 patients, each of them had two mammograms (mediolateral oblique and craniocaudal views) taken. The photos are 1024 x 1024 pixels in size and are grayscale. Along with additional data like patient age and tumor severity, the dataset also includes equivalent binary annotations showing the existence or absence of anomalies.

Teresa Araujo et al. [3] proposed a method for the classification of hematoxylin and eosin stained breast biopsy images using convolutional neural networks (CNN). The images here are classified into four classes, normal tissue, benign lesions, in situ carcinoma, and invasive carcinoma, and in two classes, carcinoma and non-carcinoma. The architecture of the network is designed to access information at a variety of scales, including nuclei and the general organization of the tissue. While normal tissue maintains its architecture and has well-organized nuclei, tissues with invasive carcinoma have a distorted architecture as well as greater nuclei density and diversity. The accuracy levels for the four classes and for carcinoma/non-carcinoma are 77.8% and 83.3%, respectively. It has a 95.6% sensitivity to cancer cases.

The image dataset used for this model was from the Bioimaging 2015 breast histology classification challenge. It has images that are high-resolution (2040 x 1536 pixels), uncompressed, and have annotated H&E stains. All the images have a magnification of 200x and a pixel size of $0.42\mu m \times 0.42\mu m$. Every image is labeled with one of the four classes, normal tissue, benign lesion, in situ carcinoma, and invasive carcinoma. The dataset consists of a separate test set of 20 photos and an expanded training set of 249 images. The four classifications in these datasets are evenly distributed. The "extended" dataset is a second test set of 16 photos that includes images with higher levels of uncertainty.

Hongdou Yao et al. [4] proposed a DL model to classify hematoxylin–eosin-stained breast biopsy images into four classes (normal tissues, benign lesions, in situ carcinomas, and invasive carcinomas). This model proposes a parallel structure consisting of CNN and RNN for image feature extraction. (In general, most of the models use CNN for extracting image features and then putting them into RNN.) The uniqueness of it is that a perceptron attention mechanism is introduced to unify the features that are extracted by CNN and RNN.

The model is tested on three different human breast cancer datasets to establish its performance.

1. **BACH2018 dataset**

A huge annotated dataset was made available by the 15th International Conference on Image Analysis and Recognition in two parts: the breast cancer histology (BACH2018) dataset and the whole-slide images (WSI). The histological mammary microscope pictures in the BACH2018 dataset were stained with hematoxylin and eosin.The training set included 100 images in each category for a uniform distribution, and the test set included 400 training images. The picture is in RGB.tiff format. The image has a resolution of 2048 x 36 pixels and a pixel size of 0.42 x 0.42 m.

2. **Bioimaging2015 dataset**

The Bioimaging 2015 breast histology classification challenge dataset included digitized hematoxylin-eosin-stained histological pictures with an image size of 2048 1536 pixels and an in-plane pixel size of 0.42 0.42 m. 36 microscopic test images and 249 microscopic training images were recorded.

3. Extended Bioimaging2015 dataset

This collection, which was an expansion of the Bioimaging2015 dataset, comprised 1319 high quality (2048 1536 pixels), uncompressed, and annotated hematoxylin and eosin color photos. The acquisition conditions were the same for all photos, which were digital and all had 200x magnification and 0.42x0.42mm pixels.

Dina A. Ragab et al. [5] proposed a system for distinguishing benign and malignant mass tumors in breast mammography pictures using computer-aided detection (CAD). The region of interest (ROI) is determined by the system using two segmentation techniques: manual segmentation and threshold-based segmentation. The authors encountered challenges because of the imbalanced datasets and the scarcity of data with ground truth annotations. They used a deep convolutional neural network (CNN) for feature extraction and then used a support vector machine (SVM) on the output of the penultimate layer of the CNN for classification. The proposed deep convolutional neural network (DCNN) and support vector machine (SVM) model achieved an accuracy of 87.2% with an AUC of 0.94 for classifying benign and malignant mass tumors in mammography images.

The authors used a publicly available dataset named "Breast Cancer Histopathological Database" (BreakHis) for training and testing their model. This dataset consists of 7,909 breast histopathological images of size 700 × 460 pixels, which are divided into two classes: benign and malignant. The benign class contains 3,905 images, while the malignant class contains 3,904 images. The images were obtained from 82 patients, and each image was annotated by an expert pathologist to ensure accuracy of the ground truth labels. The dataset is further divided into two subsets: the training set (4,015 images) and the testing set (3,894 images).

R. Zemouri et al. [6] proposed a model based on a deep constructive neural network that deals with the Breast Cancer Computer Aided Diagnosis (BC-CAD). This model is built for the Recurrence Score (RS) prediction of the Oncotype DX (ODX) breast cancer. Two classifiers are built for the classification. In the first architecture, one deep neural network is constructed for each class in a "one against all" framework. One DNN is used for the three classes in the second architecture. The class 1 states low-risk with RS<18, class 2 states intermediate-risk ($18 \leq RS < 31$), and class 3 states high-risk ($RS \geq 31$). Adjuvant chemotherapy is necessary for the more aggressive (high-risk) malignancies, but hormone therapy alone is effective for the more benign (low-risk) tumors.

The 92 instances of carcinoma mammary luminal B with available Oncotype DX test results from 2012 to 2017 were obtained from the North Trevenans County Hospital and the Georges Francois Leclerc cancer center, which are both located in Dijon and Belfort, France. The age, tumor size, ganglionic status, four separate tumor grading details, the estrogen receptor (RE), the progesterone receptor (RP), and ki67 are the ten input parameters that are used to generate the recurrence score . Three classes were established, according to the RS, with class 1 representing minimal risk with 40 instances, class 2 representing intermediate risk with 38 cases, and class 3 representing high risk with 12 instances. The first half of the data set was used for the training process, and the second half for the test.

Zhantao Cao et al. [7] performed comprehensive experiments to find out the best method for tumor detection in ultrasound images. Several existing state-of-art object detection methods were evaluated. The complicated anatomy of breasts and the presence of noise in ultrasound pictures make it difficult for classic basic feature-based approaches to produce adequate results. Deep learning's latest advancements have significantly improved object detection performance, particularly for general object detection.

From individuals with breast lesions, breast ultrasonography pictures were acquired. The patients were specifically instructed to get scans using the LOGIQ E9 (GE) and IU-Elite (PHILIPS) devices in order to produce those ultrasound images. Each diagnosed image was then classified into seven categories, ranging from 0 to 6, based on the ratings provided by the BI-RADS system, where 0 indicates that more information is required, 1 is a negative finding, 2 is a benign finding, 3 is probably benign (with a less than 2% likelihood of cancer), 4 is a suspicious abnormality, 5 is highly suggestive of malignancy, and 6 is a proven malignancy. A total of 579 benign and 464 malignant cases were collected from patients.

4. Description of the dataset

The dataset that we used for building this model is BreakHis. The two primary categories of BreakHis are benign tumors and malignant cancers. A lesion is referred to as histologically benign if it does not meet any criteria for malignancy, such as significant cellular atypia, mitosis, breakdown of basement membranes, metastasis, etc. Normal benign tumors are slow-growing, confined growths that are considered "innocent." Cancer is regarded as a malignant tumor because the lesion has the ability to spread to distant areas (metastasize) and infect other structures, causing damage and eventual death.

The dataset's samples were gathered using the SOB technique, often known as an excisional biopsy or a partial mastectomy. Compared to other needle biopsy techniques, this type of operation extracts a larger sample of tissue, and it is performed in a hospital under general anesthesia.

The appearance of the tumoral cells under a microscope might categorize breast tumors as benign or malignant. Breast tissue is examined histologically to identify several tumor types and subtypes that may have varied prognoses and treatment consequences. Under a microscope, four distinct benign breast tumor forms can be distinguished: adenosis (A), fibroadenoma (F), phyllodes tumor (PT), and tubular adenomas (TA). On the other hand, there are four types of malignant tumors that are also referred to as breast cancers: carcinoma (DC), lobular carcinoma (LC), mucinous carcinoma (MC), and papillary carcinoma (PC).

The filenames for the photos of the breast tissue samples include details on the biopsy procedure, the kind of tumor, the patient's name, and the degree of magnification. The biopsy technique, such as fine-needle aspiration (FNA) or core needle biopsy (CNB), describes how the tissue sample was acquired. The filename also contains information on the type of tumor, whether benign or malignant. The name of the patient is provided for identifying needs. A closer look at the tumoral cells is made possible by the image's enlargement, which is indicated by the magnification level.

This dataset consists of 9,109 microscopic photos of breast tumor tissue taken at different magnifications (40X, 100X, 200X, and 400X) from 82 individuals. It now has 5,429 malignant samples and 2,480 benign samples (700 x 460 pixels, 3-channel RGB, 8-bit depth per channel, PNG format). In cooperation with the P&D Laboratory - Pathological Anatomy and Cytopathology, Parana, Brazil , this database was created. This database enables future benchmarking and evaluation; therefore, we think researchers will find it beneficial.

Magnification	Benign	Malignant	Total
40X	652	1370	1995
100X	644	1437	2081
200X	623	1390	2013
400X	588	1232	1820
Total images	2480	5429	7909

For the diagnosis, prognosis, and treatment of breast cancer patients, the histological classification of breast tumors is essential. This data aids medical professionals in choosing the best course of action for treatment, including surgical alternatives and other medicines like chemotherapy or radiation. Therefore, it is crucial to accurately classify breast cancers in order to provide the best patient treatment and achieve the greatest results.

5. Methodology

5.1. Data pre-processing:

Any deep learning process requires an important stage called data preparation, which ensures that the data is acceptable for the model's training. A collection of 128 × 128-pixel photos from two classes form the project's input data. Using the Keras ImageDataGenerator, load the data into memory as the initial step in data pre-processing. This library offers a simple method for reading photos from disc, applying modifications to them, and feeding the results into a deep learning model.

Once the data has been loaded, it is important to resolve any possible class imbalance that may exist in the dataset. Class imbalance, which happens when one class has noticeably more samples than the other, can cause models to perform poorly. The Synthetic Minority Over-Sampling Technique (SMOTE) is used in this study to address the issue of class imbalance. It operates by creating synthetic samples of the minority class by interpolating between existing samples, and SMOTE is a popular strategy for addressing class imbalance. SMOTE can assist in class balancing and model performance improvement by generating fresh samples. The Python imbalanced-learn module is used to implement the SMOTE method. This library offers several options for dealing with class imbalance, including different sampling methods like SMOTE. With a variety of settings that regulate the degree of oversampling, SMOTE's implementation in imbalanced-learn enables us to create synthetic samples of the minority class. We can balance the dataset's classes and enhance the model's functionality by adjusting these parameters.

5.2. Data Augmentation:

After the data has been pre-processed, approaches for data augmentation are added to the training data to broaden the range of images the model may draw on. This enhances the model's capacity to generalize to new data and prevents overfitting. The class Keras ImageDataGenerator is utilized in this project to do data augmentation. Each image is subjected to random magnification, brightness changes, and horizontal flipping.

During training, the ImageDataGenerator class constructs a generator object that produces batches of enhanced photos. To specify the directory holding the training images and their related labels, use the flow_from_directory() method. While the batch_size option controls the number of photos in each batch, the target_size argument specifies the dimensions to which the images are scaled. To guarantee that images are handled in the order they arrive in the directory, the shuffle option is set to False.

The range of random zooming to be applied to the images is provided by the zoom_range argument. Here, a range of [.99, 1.01] is utilized, allowing for a 1% maximum magnification. The range of brightness modifications to be made to the photos is specified by the brightness_range argument. Here, a range of [0.8, 1.2] is utilized, allowing for a 20% brightness adjustment cap. The last step is to flip photos horizontally at random by setting the horizontal_flip option to True. Overall, by increasing the diversity and quantity of images the model can learn from, these data augmentation strategies serve to enhance the model's performance.

5.3. Data Splitting:

One of the most important steps in developing a machine learning model is dividing the preprocessed data into training, validation, and testing sets. The train_test_split function from the Python sklearn module is used in this project to divide the data into three sets.

The deep learning model is trained using a training set that includes 64% of the preprocessed data. 16% of the data are found in the validation set, which is utilized to adjust the hyperparameters of the model and avoid overfitting. Finally, 20% of the data are included in the test set, which is used to evaluate how well the model performs when applied to unobserved data. The data splitting procedure is essential for preventing overfitting to the training data and ensuring that the model generalizes successfully to new data. We can avoid overfitting and improve the performance of the model by tuning the hyperparameters using a validation set. For real-world applications, it is critical that the model be able to reliably categorize new data, and the test set evaluation makes sure of this.

Overall, the process of data splitting is essential to creating a successful deep learning model. It enables us to enhance the model's functionality and guarantee that it can correctly categorize unobserved data.

5.4. Methods:

5.4.1. Convolutional Neural Network

The proposed method for breast cancer classification is based on a convolutional neural network (CNN) architecture. The CNN model (fig. 1) consists of an input layer, followed by several convolutional layers, pooling layers, and fully connected layers.

Proposed CNN Architecture:

The input layer takes in the preprocessed image data of size 128x128 pixels and 3 color channels.

The first set of convolutional layers consists of two 2D convolutional layers, each with 16 filters, and with ReLU activation function and 'same' padding. Batch normalization is applied after the convolutional layers to reduce overfitting. This is followed by a max-pooling layer to reduce the dimensionality of the feature maps. The second set of convolutional layers consists of two 2D convolutional layers, each with 32 filters, with ReLU activation function and 'same' padding. Batch normalization is applied again to reduce overfitting. This is followed by another max-pooling layer. The third set of convolutional layers consists of three 2D convolutional layers, each with 64 filters, with ReLU activation function and 'same' padding. Batch normalization is applied after each convolutional layer, followed by a max-pooling layer. A dropout layer is added after the third set of convolutional layers to further reduce overfitting. The fourth set of convolutional layers consists of three 2D convolutional layers, each with 128 filters, with ReLU activation function and 'same' padding. Batch normalization is applied after each convolutional layer, followed by a max-pooling layer. Another dropout layer is added after the fourth set of convolutional layers. The fifth set of convolutional layers consists of three 2D convolutional layers, each with 256 filters, with ReLU activation function and 'same' padding. Batch normalization is applied after each convolutional layer, followed by a max-pooling layer. A final dropout layer is added after the fifth set of convolutional layers.

The output of the last max-pooling layer is flattened and passed through a series of fully connected layers. The first fully connected layer has 512 neurons with ReLU activation function and batch normalization. A dropout layer with 0.7 dropout rate is added to reduce overfitting. The second fully connected layer has 128 neurons with ReLU activation function and batch normalization. Another dropout layer with 0.5 dropout rate is added. The third fully connected layer has 64 neurons with ReLU activation function and batch normalization. A final dropout layer with 0.3 dropout rate is added. The output layer has a single neuron with sigmoid activation function, which outputs a probability value for the binary classification task.

The model is compiled with the Adam optimizer, binary cross-entropy loss function, and accuracy as the evaluation metric. The summary of the CNN model architecture and its parameters are printed for further inspection.

5.4.2. Transfer Learning Model with VGG16 base:

Transfer learning is a machine learning technique that involves using a pre-trained model as a starting point for a new task. In this case, we are using the VGG16 model as a base for our breast cancer classification task. The VGG16 model is a convolutional neural network that was trained on a large dataset of images and has achieved excellent results on image classification tasks.

The input layer of our model takes images of size 128x128 pixels with 3 channels (RGB). The output of this layer is passed through the VGG16 model, which consists of several convolutional

CNN

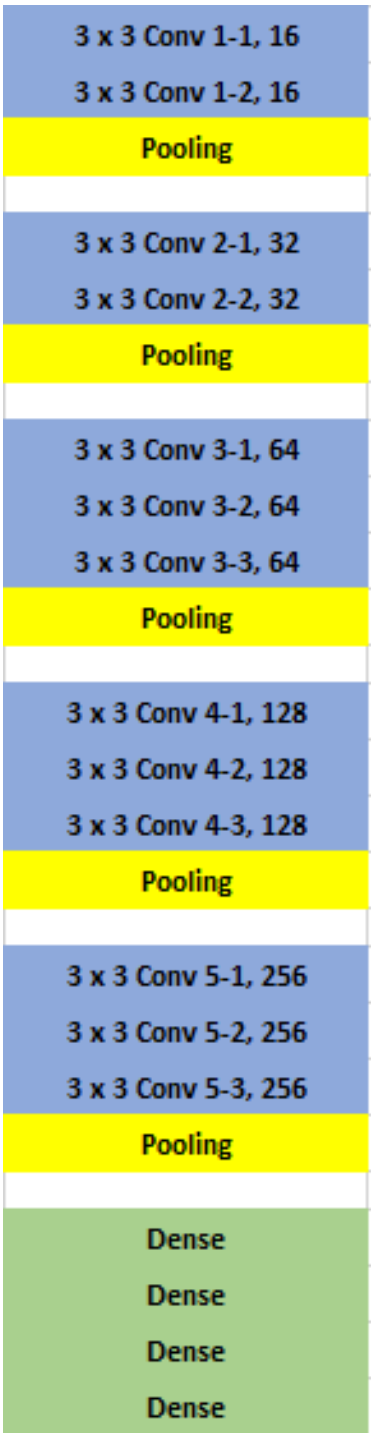


Fig. 1

ResNet50V2

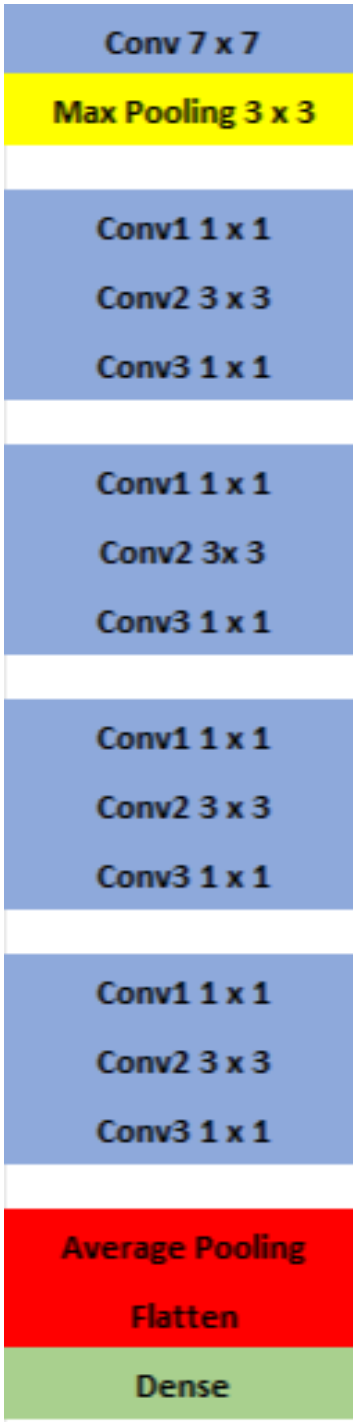


Fig. 2

and pooling layers. These layers extract features from the input image that are relevant for classification. Since the VGG16 model has already been trained on a large dataset, we freeze the weights of its layers and only train the new layers we add on top.

After the VGG16 model, we add a flatten layer that converts the output of the convolutional layers into a one-dimensional feature vector. This is followed by several dense layers, each with its own batch normalization and dropout layers to prevent overfitting. The final layer uses a sigmoid activation function to output a probability score between 0 and 1, representing the likelihood of the input image belonging to the positive (cancerous) class. Our model is trained using the binary cross-entropy loss function and the Adam optimizer. During training, we use data augmentation techniques such as random horizontal flips and rotations to increase the diversity of the training data and improve the generalization of our model. We use a validation set to monitor the performance of our model during training and prevent overfitting.

Overall, our transfer learning model with VGG16 base architecture consists of a series of convolutional and dense layers with batch normalization and dropout layers to prevent overfitting. By using a pre-trained model as a base, we are able to leverage the knowledge learned from a large dataset and achieve good performance on our breast cancer classification task with less training data and computational resources.

5.4.3. Transfer Learning Model with ResNet50V2:

Transfer learning is a technique that involves using a pre-trained model as a starting point for a new task. This approach has become increasingly popular in the field of machine learning because it enables us to utilize the knowledge and expertise gained from a large-scale dataset to solve new tasks more efficiently. We are using the ResNet50V2 model as a base for our breast cancer classification task.

The ResNet50V2 (fig. 2) model is a deep convolutional neural network that has been pre-trained on the ImageNet dataset, which contains millions of images. This pre-training enables the model to extract high-level features from images that can be used to perform other computer vision tasks. By using the pre-trained ResNet50V2 model as a backbone, we can take advantage of its powerful feature extraction capabilities while significantly reducing the time and resources required to train a deep neural network from scratch.

To adapt the ResNet50V2 model for our specific task, we freeze the pre-trained layers and add new layers to the model. These new layers include fully connected layers with dropout and batch normalization, as well as an output layer with a sigmoid activation function. The fully connected layers enable the model to learn specific features relevant to our breast cancer classification task,

while the dropout and batch normalization layers improve the model's generalization performance.

The addition of these new layers also allows us to fine-tune the pre-trained ResNet50V2 model to improve its performance on our specific task. By adjusting the weights of the pre-trained layers during training, we can fine-tune the model to better fit the features of our dataset. This fine-tuning process enables the model to learn more specific features relevant to our task, ultimately leading to improved classification performance.

Overall, transfer learning and the ResNet50V2 architecture provide a powerful framework for solving a variety of computer vision tasks, including breast cancer classification. By leveraging the pre-trained ResNet50V2 model, we can take advantage of its powerful feature extraction capabilities, reduce training time and resources, and improve the overall performance of our deep neural network model.

6. Experimental Results

6.1. Model Evaluation of Convolutional Neural Network:

The CNN model correctly classified the majority of the breast cancer histopathology photos on the test set, with an accuracy of 88.50%. In addition, the model showed balanced performance in properly classifying both benign and malignant breast cancer lesions, with average precision, recall, and F1 scores of 88.63%, 88.50%, and 88.48%, respectively.

We also calculated the AUC score and plotted the ROC curve (shown in Fig. 3) to further assess the model's performance. The AUC value summarizes the overall performance of the classifier, while the ROC curve illustrates the trade-off between true positive rate and false positive rate for various threshold settings. Our model successfully distinguished between benign and malignant breast tissues with an AUC score of 0.96.

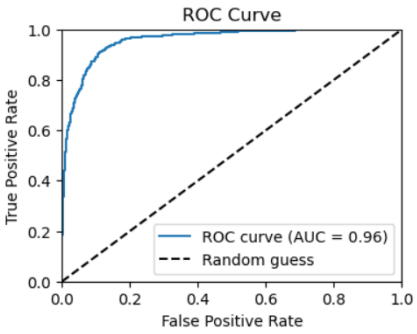


Fig. 3

The precision-recall curve (shown in Fig. 4) depicts the trade-off between precision and recall at various threshold levels. With a precision and recall of 88.63% and 88.50%, respectively, our model was able to accurately identify the majority of malignant breast lesions while reducing false positives.

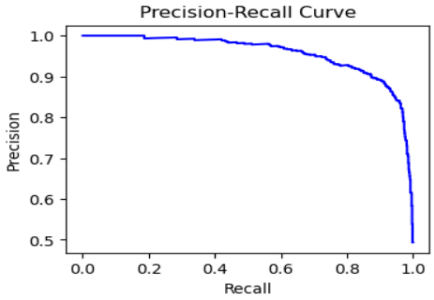


Fig. 4

The confusion matrix (shown in Fig. 5) displays the number of true positive, false positive, true negative, and false negative predictions made by the model. It was the last visualization we created. According to the confusion matrix, our algorithm successfully identified 751 benign and 672 malignant breast lesions while misidentifying 117 benign and 68 malignant breast lesions.

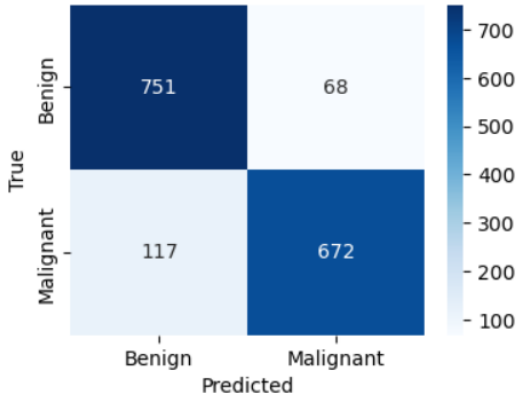


Fig. 5

Overall, our model performed well when determining whether breast lesions were benign or malignant. Exploring various architectures, hyperparameters, and preprocessing methods could lead to further advancements.

6.2. Model Evaluation of Transfer Learning:

6.2.1. Using VGG16:

Using the VGG16 architecture, we created a transfer learning model that successfully classified histopathology images of breast cancer as benign or malignant with an astounding 86.57% accuracy. 80% of the original dataset was used to train this model, while the remaining 20% was used to validate it. The model was trained using the Adam optimizer over 50 epochs using a binary cross-entropy loss function.

The VGG16 design, a popular convolutional neural network architecture for image classification applications, served as the foundation for our transfer learning model. To avoid overfitting for our dataset, we used pre-trained weights from the ImageNet dataset and frozen the weights of the VGG16 layers. In order to increase generalization and decrease overfitting, we next added a number of fully connected layers to the network, followed by batch normalization and dropout layers.

With the help of a variety of indicators, such as a classification report, the ROC curve (shown in Fig. 6), and the precision-recall curve (shown in Fig. 7), we assessed the performance of our transfer learning model. According to the classification report, our model had a good level of recall and precision for both benign and malignant classifications. Our model's AUC score of 0.93 on the ROC curve indicated a decent balance between sensitivity and specificity. The accuracy-recall curve additionally demonstrated that our model had excellent precision and recall for both classes.

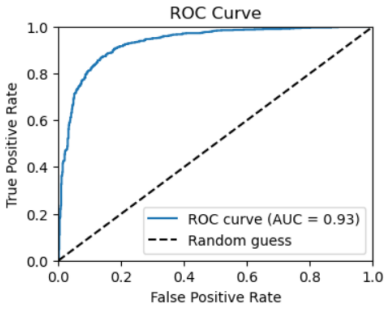


Fig. 6

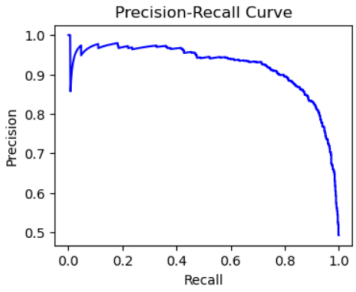


Fig. 7

Overall, the classification of breast cancer histopathology images as benign or malignant using our transfer learning model yielded encouraging results. This model has the potential to be employed as a tool for automated diagnosis of breast cancer, which may save lives by enabling earlier diagnosis and treatment of the condition.

6.2.2. Using ResNet50V2:

ResNet50V2 is a potent convolutional neural network design that has been successfully applied to highly accurate picture categorization tasks. Using the ResNet50V2 pre-trained model, we built a binary image classifier in this project that uses transfer learning to distinguish between benign and malignant breast cancer tumors.

We created a sequential model by adding a flatten layer to flatten the output of the pre-trained model, three dense layers with ReLU activation, batch normalization, and dropout layers for regularization, and then loading and freezing the layers of the pre-trained ResNet50V2 model. Binary output was generated by the final dense layer using the sigmoid activation function.

The binary cross-entropy loss function and Adam optimizer were used to create the model. The test set was used to assess the model's accuracy, which was determined to be 86.63%, indicating that the model did a good job of identifying images of breast cancers as benign or malignant.

We also assessed the ResNet50V2 model's performance using classification measures like precision-recall (shown in Fig. 8), F1 score, and AUC-ROC (shown in Fig. 9). It was discovered that the model had average precision, recall, and F1 scores of 86.63%, 86.63%, and 86.62%,

respectively. Additionally, the model performed exceptionally well in terms of differentiating between benign and malignant tumors, with an AUC-ROC score of 0.96. The precision-recall curve showed that even at high recall values, the model's precision remained high. The model had high true positive rates and low false positive rates, according to the ROC curve.

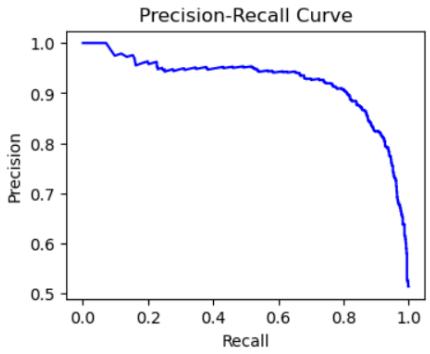


Fig. 8

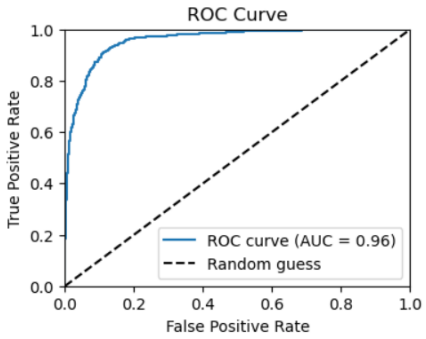


Fig. 9

In conclusion, our findings show that the ResNet50V2 model is a very accurate model with great performance in terms of precision, recall, F1 score, and AUC-ROC for categorizing breast tumor images as benign or malignant.

6.3. Comparison between all the models

Three pre-trained convolutional neural network (CNN) models were examined for performance: CNN, VGG16, and ResNet50V2. The test dataset was utilized to assess the models, and the following outcomes were attained:

Model	Loss	Accuracy
CNN	0.29	0.88
VGG16	0.50	0.87
ResNet50V2	0.71	0.87

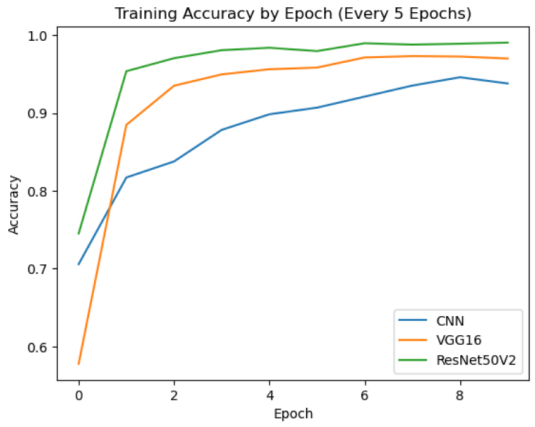
The table shows thatCNN has the best accuracy (0.88), and lowest loss (0.29), and is followed by VGG16 with accuracy (0.87), loss (0.50), and ResNet50V2 with accuracy (0.87) and loss (0.71).

For each model, we also determined the average F-1 score, average precision, and average recall. The following table displays the results:

Model	Avg Precision	Avg Recall	Avg F-1 Score
CNN	88.625755	88.495025	88.477759
VGG16	86.586633	86.567164	86.568328
ResNet50V2	86.657081	86.629353	86.622235

The CNN model achieved the highest avg precision, avg recall and avg F-1 score, followed by ResNet50V2 model, then followed by VGG16. These results suggest that the CNN is the most accurate and robust model among the three pre-trained CNN models we evaluated.

We also plotted the training accuracy (Fig. 10), and loss history (Fig 11) for each model.



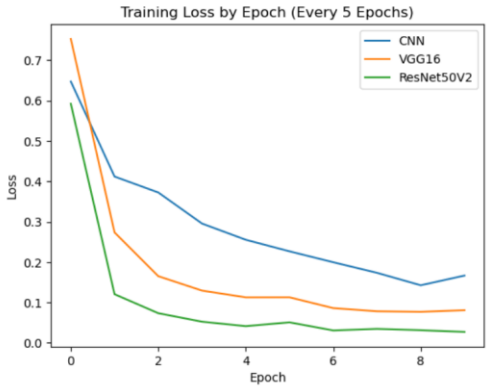


Fig. 10 **Fig. 11**

The ResNet50V2 model, followed by the VGG16 model and then the CNN model, had the highest training accuracy and the lowest training loss, as seen in the figures.

Overall, the results suggest that the CNN model is the most accurate and robust model among the three pre-trained CNN models we evaluated.

7. Conclusion

In this paper, a deep learning model for improving the classification results on the BreakHis dataset was proposed. The purpose of this model is to help medical doctors in BC detection and diagnosis. The BreakHis images were divided into two different classes, benign and malignant. The original BreakHis dataset was pre-processed for resizing the images to a fixed resolution, converting the images into particular size, and then dividing them into training and testing sets. The data augmentation concept was also proposed for increasing the size of dataset to enhance the performance of the Proposed Models. Then, the freezing and fine-tuning strategies were used to improve the classification accuracy of the mentioned dataset. The results show that the suggested method is successful in correctly categorizing breast cancer photos, with an accuracy rate of 88%. The results of this study have important ramifications for the creation of automated methods for diagnosing breast cancer that can help medical practitioners make precise and prompt diagnosis. The CNN model appears to have achieved the highest accuracy of the three tested models, according to the experimental results based on the performance metrics of Average Precision, Average Recall, and Average F1-Score, with an average precision of 88.63%, an average recall of 88.50%, and an average F1-score of 88.48%. With an average precision of 86.59%, average recall of 86.57%, and average F1-score of 86.57%, the VGG16 model showed somewhat poorer results. Similar results were obtained with the ResNet50V2 model, which had average precision, recall, and F1-scores of 86.66%, 86.63%, and 86.62% respectively.

References

[1] Z. Han, . B. W. . Y. Z. . Y. Y. . K. L. and . S. L. , "Breast Cancer Multi-classification from Histopathological Images with Structured Deep Learning Model," *Scientific Reports,* 2017.

[2] S. Khan, N. I. . Z. J. I. U. D. and J. J. P. C. R. , "A novel deep learning based framework for the detection and classification of breast cancer using transfer learning," *Pattern Recognition Letters,* vol. 125, pp. 1-6, 2019.

[3] T. Araújo, G. A. E. C. J. R. P. A. C. E. A. P. and A. C. , "Classification of breast cancer histology images using Convolutional Neural Networks," *Plos One,* 2017.

[4] H. Yao, X. Z. X. Z. and S. L. , "Parallel Structure Deep Neural Network Using CNN and RNN with an Attention Mechanism for Breast Cancer Histology Image Classification," *MDPI,* vol. 11, no. 12, 2019.

[5] D. A. Ragab, M. S. S. M. and J. R. , "Breast cancer detection using deep convolutional neural networks and support vector machines," *BIOINFORMATICS AND GENOMICS,* 2019.

[6] R. Zemouri, . N. O. B. M. C. D. L. A. . N. Z. and F. F. , "Constructive Deep Neural Network for Breast Cancer Diagnosis," *IFAC-PapersOnLine,* vol. 51, no. 27, pp. 98-103, 2018.

[7] Z. Cao, L. D. G. Y. T. Y. . Q. C. H. F. and Y. X. , "Breast Tumor Detection in Ultrasound Images".

Appendix:

```
import keras
import matplotlib.pyplot as plt
import numpy as np
import os
import pandas as pd
import pathlib
import seaborn as sns
import tensorflow as tf
import tensorflow_addons as tfa

from keras.utils import plot_model
from imblearn.over_sampling import SMOTE
from keras.callbacks import ModelCheckpoint, EarlyStopping
from keras.layers import BatchNormalization, Conv2D, Conv2DTranspose, Dense, Dropout,
Flatten, GlobalAveragePooling2D, Input, MaxPooling2D
from keras.models import Model, Sequential
from keras.preprocessing.image import ImageDataGenerator
from sklearn.metrics import classification_report, confusion_matrix, precision_recall
_curve, roc_curve, roc_auc_score
from tensorflow.keras.applications import ResNet50V2, VGG16, InceptionV3
from tensorflow.keras.preprocessing.image import ImageDataGenerator
from keras.utils import load_img,img_to_array
```

In [2]:

```
folder = '/kaggle/input/breakhis/BreaKHis_v1/BreaKHis_v1/histology_slides/breast'
folder_path = pathlib.Path(folder)
```

In [3]:

```
linkcode
photo = load_img('/kaggle/input/breakhis/BreaKHis_v1/BreaKHis_v1/histology_slides/bre
ast/benign/SOB/adenosis/SOB_B_A_14-22549AB/100X/SOB_B_A-14-22549AB-100-001.png')
print(photo)
photo
```

In [4]:

```
IMG_SIZE = 128
DIM = (IMG_SIZE, IMG_SIZE)

ZOOM = [.99, 1.01]
BRIGHT_RANGE = [0.8, 1.2]
HORZ_FLIP = True
FILL_MODE = "constant"
DATA_FORMAT = "channels_last"
```

In [5]:

```
train_generator = ImageDataGenerator(rescale = 1./255, brightness_range=BRIGHT_RA
NGE, zoom_range=ZOOM,
                                     data_format=DATA_FORMAT, fill_mode=FILL_MODE
, horizontal_flip=HORZ_FLIP)
train_data_gen = train_generator.flow_from_directory(directory=folder, target_siz
e=DIM, batch_size=6500, shuffle=False)
```

In [6]:

```
train_data, train_labels = train_data_gen.next()
```

In [7]:

```
sm = SMOTE(random_state=42)

train_data, train_labels = sm.fit_resample(train_data.reshape(-1, IMG_SIZE * IMG_SIZE
* 3), train_labels)

print(train_data.shape, train_labels.shape)
```

In [8]:

```
train_data = train_data.reshape(-1, IMG_SIZE, IMG_SIZE, 3)
print(train_data.shape, train_labels.shape)
```

In [9]:

```
train_labels = train_labels.reshape(-1, 1)
```

In [10]:

```
linkcode
for i in range(2):
    plt.subplot(2,2,1+i)
    plt.title(train_labels[i])
```

In [11]:

```
from sklearn.model_selection import train_test_split
```

In [12]:

```
linkcode
train_data, test_data, train_labels,test_labels = train_test_split(train_data, train_
labels, test_size = 0.2, random_state=42)

train_data, val_data, train_labels,val_labels = train_test_split(train_data, train_la
bels, test_size = 0.2, random_state=42)
```

in [13]:

```
from keras.layers import MaxPool2D

act = 'relu'
IMAGE_SIZE = (128, 128)

cnn_model = Sequential([
Input(shape=(IMAGE_SIZE[0], IMAGE_SIZE[1], 3)),
Conv2D(16, 3, activation=act, padding='same'),
Conv2D(16, 3, activation=act, padding='same'),
BatchNormalization(),
MaxPool2D(),
Conv2D(32, 3, activation=act, padding='same'),
Conv2D(32, 3, activation=act, padding='same'),
BatchNormalization(),
MaxPool2D(),
Conv2D(64, 3, activation=act, padding='same'),
Conv2D(64, 3, activation=act, padding='same'),
Conv2D(64, 3, activation=act, padding='same'),
BatchNormalization(),
```

```
MaxPool2D(),
Dropout(0.2),
Conv2D(128, 3, activation=act, padding='same'),
Conv2D(128, 3, activation=act, padding='same'),
Conv2D(128, 3, activation=act, padding='same'),
BatchNormalization(),
    MaxPool2D(),
Dropout(0.2),
Conv2D(256, 3, activation=act, padding='same'),
Conv2D(256, 3, activation=act, padding='same'),
Conv2D(256, 3, activation=act, padding='same'),
BatchNormalization(),
MaxPool2D(),
Dropout(0.2),
Flatten(),
Dense(512, activation=act),
BatchNormalization(),
Dropout(0.7),
Dense(128, activation=act),
BatchNormalization(),
Dropout(0.5),
Dense(64, activation=act),
BatchNormalization(),
Dropout(0.3),
Dense(1, activation='sigmoid')
], name='cnn_model')
```

In [14]:

```
# Compile the model
cnn_model.compile(optimizer='adam', loss='binary_crossentropy', metrics=['accurac
y'])

cnn_model.summary()
```

In [15]:

```
plot_model(cnn_model)
```

In [[16]:

```
tf.keras.backend.set_image_data_format('channels_first')
```

In [17]:

```
# Fit the model to the training data
history = cnn_model.fit(train_data, train_labels, epochs=50, verbose=1)

# Store accuracy and loss history for each epoch
cnn_accuracy_history = history.history['accuracy']
cnn_loss_history = history.history['loss']
```

In [18]:

```
loss, accuracy = cnn_model.evaluate(test_data, test_labels)
print("Accuracy: {:.2f}%".format(accuracy*100))
```

In [19]:

```
linkcode
y_pred = np.round(cnn_model.predict(test_data)).astype(int)
```

```
target_names = ['Benign', 'Malignant']
report = classification_report(test_labels, y_pred, target_names=target_names, output
_dict=True)

for target_name in target_names:
    precision = report[target_name]['precision']
    recall = report[target_name]['recall']
    f1_score = report[target_name]['f1-score']

# Average precision, recall, and F1-score
avg_precision = report['weighted avg']['precision']
avg_recall = report['weighted avg']['recall']
avg_f1_score = report['weighted avg']['f1-score']

print('Average Precision: {:.2f}%'.format(avg_precision*100))
print('Average Recall: {:.2f}%'.format(avg_recall*100))
print('Average F1 Score: {:.2f}%'.format(avg_f1_score*100))

cnn_precision = avg_precision
cnn_recall = avg_recall
cnn_f1score = avg_f1_score
```

In [20]:

```
# Get predicted probabilities for test set
y_pred_prob = cnn_model.predict(test_data)

# Get false positive rate, true positive rate, and thresholds
fpr, tpr, thresholds = roc_curve(test_labels, y_pred_prob)

# Compute AUC score
roc_auc = roc_auc_score(test_labels, y_pred_prob)

# Plot ROC curve
plt.figure(figsize=(4,3))
plt.plot(fpr, tpr, label='ROC curve (AUC = {:.2f})'.format(roc_auc))
plt.plot([0, 1], [0, 1], 'k--', label='Random guess')
plt.xlim([0, 1])
plt.ylim([0, 1])
plt.xlabel('False Positive Rate')
plt.ylabel('True Positive Rate')
plt.title('ROC Curve')
plt.legend(loc='lower right')
plt.show()
```

In [21]:

```
y_scores = cnn_model.predict(test_data)
precision, recall, _ = precision_recall_curve(test_labels, y_scores)

plt.figure(figsize=(4,3)) # set figure size
plt.plot(recall, precision, color='b')
plt.xlabel('Recall')
plt.ylabel('Precision')
plt.title('Precision-Recall Curve')
```

```
plt.show()
51/51 [==============================] - 0s 9ms/step
```

In [22]:

```
# your code here
cm = confusion_matrix(test_labels, y_pred)
plt.figure(figsize=(4, 3))
sns.heatmap(cm, annot=True, fmt='g', cmap='Blues', xticklabels=target_names, yticklab
els=target_names)
plt.xlabel('Predicted')
plt.ylabel('True')
plt.show()
```

In [23]:

```
from keras import backend as K

K.set_image_data_format('channels_last')
vgg_model = VGG16(weights='imagenet', include_top=False, input_shape=(128, 128, 3))
```

In [24]:

```
for layer in vgg_model.layers:
    layer.trainable = False
```

In [25]:

```
vgg16_model = Sequential([
    vgg_model,
    Flatten(),
    BatchNormalization(),
    Dense(512, activation='relu'),
    BatchNormalization(),
    Dropout(0.5),
    Dense(256, activation='relu'),
    BatchNormalization(),
    Dropout(0.5),
    Dense(128, activation='relu'),
    BatchNormalization(),
    Dropout(0.5),
    Dense(64, activation='relu'),
    Dropout(0.5),
    BatchNormalization(),
    Dense(1, activation='sigmoid')
], name="vgg16_model")
```

In [26]:

```
vgg16_model.compile(optimizer='adam', loss='binary_crossentropy', metrics=['accuracy'
])
vgg16_model.summary()
```

In [27]:

```
plot_model(vgg16_model)
```

In [28]:

```
# Fit the model to the training data
vgg16_history = vgg16_model.fit(train_data, train_labels, epochs=50, verbose=1)
```

```
# Store accuracy and loss history for each epoch
vgg16_accuracy_history = vgg16_history.history['accuracy']
vgg16_loss_history = vgg16_history.history['loss']
```

In [29]:

```
loss, accuracy = vgg16_model.evaluate(test_data, test_labels)
print("Accuracy: {:.2f}%".format(accuracy*100))
```

In [30]:

```
y_pred = np.round(vgg16_model.predict(test_data)).astype(int)

target_names = ['Benign', 'Malignant']
report = classification_report(test_labels, y_pred, target_names=target_names, output
_dict=True)

for target_name in target_names:
    precision = report[target_name]['precision']
    recall = report[target_name]['recall']
    f1_score = report[target_name]['f1-score']

# Average precision, recall, and F1-score
avg_precision = report['weighted avg']['precision']
avg_recall = report['weighted avg']['recall']
avg_f1_score = report['weighted avg']['f1-score']

print('Average Precision: {:.2f}%'.format(avg_precision*100))
print('Average Recall: {:.2f}%'.format(avg_recall*100))
print('Average F1 Score: {:.2f}%'.format(avg_f1_score*100))

vgg16_precision = avg_precision
vgg16_recall = avg_recall
vgg16_f1score = avg_f1_score
```

In [31]:

```
# Get predicted probabilities for test set
y_pred_prob_tl = vgg16_model.predict(test_data)

# Get false positive rate, true positive rate, and thresholds
fpr, tpr, thresholds = roc_curve(test_labels, y_pred_prob_tl)

# Compute AUC score
roc_auc = roc_auc_score(test_labels, y_pred_prob_tl)

# Plot ROC curve
plt.figure(figsize=(4,3))
plt.plot(fpr, tpr, label='ROC curve (AUC = {:.2f})'.format(roc_auc))
plt.plot([0, 1], [0, 1], 'k--', label='Random guess')
plt.xlim([0, 1])
plt.ylim([0, 1])
plt.xlabel('False Positive Rate')
plt.ylabel('True Positive Rate')
plt.title('ROC Curve')
plt.legend(loc='lower right')
```

```
plt.show()
```

In [32]:

```
y_scores = vgg16_model.predict(test_data)
precision, recall, _ = precision_recall_curve(test_labels, y_scores)

plt.figure(figsize=(4,3)) # set figure size
plt.plot(recall, precision, color='b')
plt.xlabel('Recall')
plt.ylabel('Precision')
plt.title('Precision-Reca
```

In [33]:

```
# Load the pre-trained ResNet50V2 model
resnet50v2 = ResNet50V2(include_top=False, weights='imagenet', input_shape=(128, 128,
3))

# Freeze the layers in the pre-trained model
for layer in resnet50v2.layers:
    layer.trainable = False
```

In [34]:

```
ResNet50V2_model = Sequential([
    resnet50v2,
    Flatten(),
    Dense(512, activation='relu'),
    BatchNormalization(),
    Dropout(0.5),
    Dense(256, activation='relu'),
    BatchNormalization(),
    Dropout(0.5),
    Dense(1, activation='sigmoid')
], name="ResNet50V2_model")
```

In [35]:

```
ResNet50V2_model.compile(optimizer='adam', loss='binary_crossentropy', metrics=['accu
racy'])

ResNet50V2_model.summary()
```

In [36]:

```
plot_model(ResNet50V2_model)
```

In [37]:

```
ResNet50V2_model_history = ResNet50V2_model.fit(train_data, train_labels, epochs=
50, verbose=1)

# Store accuracy and loss history for each epoch
ResNet_accuracy_history = ResNet50V2_model_history.history['accuracy']
ResNet_loss_history = ResNet50V2_model_history.history['loss']
```

In [38]:

```
loss, accuracy = ResNet50V2_model.evaluate(test_data, test_labels)
print("Accuracy: {:.2f}%".format(accuracy*100))
```

In [39]:

```
y_pred = np.round(ResNet50V2_model.predict(test_data)).astype(int)

target_names = ['Benign', 'Malignant']
report = classification_report(test_labels, y_pred, target_names=target_names, output
_dict=True)

for target_name in target_names:
    precision = report[target_name]['precision']
    recall = report[target_name]['recall']
    f1_score = report[target_name]['f1-score']

# Average precision, recall, and F1-score
avg_precision = report['weighted avg']['precision']
avg_recall = report['weighted avg']['recall']
avg_f1_score = report['weighted avg']['f1-score']

print('Average Precision: {:.2f}%'.format(avg_precision*100))
print('Average Recall: {:.2f}%'.format(avg_recall*100))
print('Average F1 Score: {:.2f}%'.format(avg_f1_score*100))

ResNet_precision = avg_precision
ResNet_recall = avg_recall
ResNet_f1score = avg_f1_score
```

In [40]:

```
# Get predicted probabilities for test set
y_pred = ResNet50V2_model.predict(test_data)

# Get false positive rate, true positive rate, and thresholds
fpr, tpr, thresholds = roc_curve(test_labels, y_pred_prob)

# Compute AUC score
roc_auc = roc_auc_score(test_labels, y_pred_prob)

# Plot ROC curve
plt.figure(figsize=(4,3))
plt.plot(fpr, tpr, label='ROC curve (AUC = {:.2f})'.format(roc_auc))
plt.plot([0, 1], [0, 1], 'k--', label='Random guess')
plt.xlim([0, 1])
plt.ylim([0, 1])
plt.xlabel('False Positive Rate')
plt.ylabel('True Positive Rate')
plt.title('ROC Curve')
plt.legend(loc='lower right')
plt.show()
```

In [41]:

```
y_scores = ResNet50V2_model.predict(test_data)
precision, recall, _ = precision_recall_curve(test_labels, y_scores)

plt.figure(figsize=(4,3)) # set figure size
plt.plot(recall, precision, color='b')
plt.xlabel('Recall')
```

```
plt.ylabel('Precision')
plt.title('Precision-Recall Curve')
plt.show()
```

In [42]:

```
# Evaluate the CNN model
cnn_loss, cnn_accuracy = cnn_model.evaluate(test_data, test_labels)

# Evaluate the VGG16 model
vgg16_loss, vgg16_accuracy = vgg16_model.evaluate(test_data, test_labels)

# Evaluate the ResNet50V2 model
resnet_loss, resnet_accuracy = ResNet50V2_model.evaluate(test_data, test_labels)

print('+-----------------------------------------------------+')
print('|{"Model Evaluation Results"} |   {"loss"} | {"accuracy"}             |')
print('+-----------------+---------------+---------------------+')
print(f'|    {"CNN Model":<15} |    {cnn_loss:.2f}      |        {cnn_accuracy:.2
f}        |')
print('+-----------------+---------------+---------------------+')


print(f'|   {"VGG16 Model":<15} |    {vgg16_loss:.2f}      |        {vgg16_accura
cy:.2f}        |')
print('+-----------------+---------------+---------------------+')


print(f'| {"ResNet50V2 Model":<15} |    {resnet_loss:.2f}      |        {resnet_a
ccuracy:.2f}         |')
print('+-----------------+---------------+---------------------+')
```

In [43]:

```
# Extract the accuracy history for each model every 5 epochs
cnn_history = cnn_accuracy_history[::5]
vgg16_history = vgg16_accuracy_history[::5]
resnet_history = ResNet_accuracy_history[::5]

# Plot the accuracy history for each model
plt.plot(cnn_history, label='CNN')
plt.plot(vgg16_history, label='VGG16')
plt.plot(resnet_history, label='ResNet50V2')

# Set the plot title and axis labels
plt.title('Training Accuracy by Epoch (Every 5 Epochs)')
plt.xlabel('Epoch')
plt.ylabel('Accuracy')

# Add a legend to the plot
plt.legend()

# Show the plot
plt.show()
```

In [44]:

```
# Extract the loss history for each model every 5 epochs
cnn_history = cnn_loss_history[::5]
vgg16_history = vgg16_loss_history[::5]
resnet_history = ResNet_loss_history[::5]

# Plot the loss history for each model
plt.plot(cnn_history, label='CNN')
plt.plot(vgg16_history, label='VGG16')
plt.plot(resnet_history, label='ResNet50V2')

# Set the plot title and axis labels
plt.title('Training Loss by Epoch (Every 5 Epochs)')
plt.xlabel('Epoch')
plt.ylabel('Loss')

# Add a legend to the plot
plt.legend()

# Show the plot
plt.show()
```

In [45]:

```
# Create a dictionary to store the metrics
metrics_dict = {
    'Model': ['CNN', 'VGG16', 'ResNet50V2'],
   'Avg Precision': [cnn_precision*100, vgg16_precision*100, ResNet_precision*100
],
        'Avg Recall': [cnn_recall*100, vgg16_recall*100, ResNet_recall*100],
        'Avg F1-Score': [cnn_f1score*100, vgg16_f1score*100, ResNet_f1score*100]
}

# Create a pandas DataFrame from the dictionary
metrics_df = pd.DataFrame(metrics_dict)

# Print the DataFrame
print(metrics_df)
```