REPUBLIC OF TURKEY
YILDIZ TECHNICAL UNIVERSITY
DEPARTMENT OF COMPUTER ENGINEERING

CLASSIFICATION OF BREAST CANCER IMAGES USING DEEP LEARNING METHODS

15011902 — Emre ÇELİK
15011091 — Kamil BAŞKUT

SENIOR PROJECT

Advisor
Assoc. Prof. Gökhan BİLGİN

January, 2021

ACKNOWLEDGEMENTS

First of all, we would like to thank the Yıldız Technical University Computer Engineering Department for giving such an opportunity for this project.

We would like to thank our advisor, Assoc. Prof. Gökhan BİLGİN, for giving us the opportunity to do research and for providing us with invaluable guidance throughout this research. It was a great privilege and honor to work and study under his guidance. We are extremely grateful for what he has to offer us.

Thanks to everyone who provided access to the methods and packages used in this project and supported the development of open-source code.

Finally, we would like to thank our family for being supportive and tolerant throughout our lives.

Emre ÇELİK
Kamil BAŞKUT

TABLE OF CONTENTS

LIST OF ABBREVIATIONS

BACH	Breast Cancer Histology
CNN	Convolutional Neural Network
CNTK	The Microsoft Cognitive Toolkit
CPU	Central Processing Unit
DCNN	Deep Convolutional Neural Network
FLOPS	Floating Point Operations Per Second
GPU	Graphics Processing Unit
H&E	Hematoxylin and Eosin
ICIAR	The International Conference on Image Analysis and Recognition
RAM	Random Access Memory
SGD	Stochastic Gradient Descent
SVM	Support Vector Machine

LIST OF FIGURES

LIST OF TABLES

ABSTRACT

Classification of Breast Cancer Images Using Deep Learning Methods

Emre ÇELİK
Kamil BAŞKUT

Department of Computer Engineering
Senior Project

Advisor: Assoc. Prof. Gökhan BİLGİN

The popularity of deep learning is increasing day by day in the medical world. In this project, breast cancer classification, which is important to detect at an early stage, was made using Transfer Learning. The goal of this project is to accurately classify breast cancer images with a high success rate. BACH and Bioimaging data sets were used in the project. The BACH dataset was used in the training phase, and the Bioimaging data set was used in the testing phase. The models used in transfer learning were selected from the Keras library. The data set pictures are divided into patches of 256 × 256, 128 × 128, 75 × 75 sizes from the places where the cell density is high. The bottleneck feature, which provides higher performance in low image numbers, is used. The number of epochs was determined as 16, and it was aimed to investigate the effect of patch size on the accuracy rate between models. The accuracy rates and confusion matrix of the models used in the project are shown with the interface. When we give four randomly selected images to the model on the prediction page, the result is shown with the pictures.

Keywords: CNN, transfer learning, breast cancer, ICIAR, BACH, Bioimaging

ÖZET

Göğüs Kanseri Görüntülerinin Derin Öğrenme Yöntemleri ile Sınıflandırılması

Emre ÇELİK
Kamil BAŞKUT

Bilgisayar Mühendisliği Bölümü
Bitirme Projesi

Danışman: Doç. Dr. Gökhan BİLGİN

Derin öğrenmenin popülerliği tıp dünyasında her geçen gün artmaktadır. Bu projede erken aşamada tespiti önemli olan göğüs kanseri sınıflandırılması, Transfer Learning kullanılarak yapılmıştır. Bu projenin hedefi, göğüs kanseri görüntülerinin yüksek başarı oranı ile doğru şekilde sınıflandırılmasıdır. Projede BACH ve Bioimaging veri setleri kullanılmıştır. Eğitim aşamasında BACH veriseti kullanılmış olup test aşamasında Bioimaging veri seti kullanılmıştır. Transfer learning yapılırken kullanılan modeller Keras kütüphanesi içinden seçilmiştir. Veri setindeki resimler, hücre yoğunluğunun fazla olduğu yerlerinden 256×256, 128×128, 75×75 boyutlarında patchlere ayrılmıştır. Düşük resim sayılarında daha yüksek başarım sağlayan bottleneck özelliği kullanılmıştır. Epoch sayısı 16 olarak belirlenmiş olup patch boyutunun modeller arasındaki doğruluk oranına etkisinin araştırılması hedeflenmiştir. Yapılmış olan arayüz ile projede kullanılan modellerin doğruluk oranları ve confusion matrisi gösterilmektedir. Tahmin sayfasında ise rastgele seçilen dört resmi modele verdiğimizde bize verdiği sonuc resimler ile birlikte gösterilmektedir.

Anahtar Kelimeler: CNN, transfer learning, göğüs kanseri, ICIAR, BACH, Bioimaging

1
Introduction

Breast cancer is considered the most commonly observed cancer among women, where it reaches approximately 2.1 million women per annum and leads to an enormous death rate. About 627,000 women [1] with breast cancer died in the year 2018. Breast cancer disease has increased amongst women in more progressive areas, and it is growing in every place globally.

To decrease breast cancer rates, early detection remains a critical part. Early detection approaches focus on providing a suitable cancer treatment method by reducing difficulties in upkeep and refining access to actual detection services. This study includes research on the early detection of breast cancer with deep learning methods.

In this study, we aim to develop a fully automatic, deep learning-based method using descriptor features extracted by Deep Convolutional Neural Network (DCNN) models and pooling operation for the classification of hematoxylin and eosin stain (H&E) histological breast cancer images provided as a part of the International Conference on Image Analysis and Recognition (ICIAR) 2018 Grand Challenge on BreAst Cancer Histology (BACH) and Bioimaging 2015 clinical challenge images. Different data augmentation methods are applied to optimize DCNN performance.

2
Literature Review

In [2], Rakhlin et al. proposed a deep learning-based method for the classification of H&E stained breast tissue images. For each image, 20 crops of 400×400 pixels and 650×650 were extracted. Then, pre-trained ResNet-50, InceptionV3, and VGG-16 networks were used as feature extractors. Extracted features were combined through 3-norm pooling into a single feature vector. A LightGBM classifier with 10-fold cross-validation was used to classify extracted deep features. That method achieved an average accuracy of 87.2 ± 2.6% across all folds for the classification of the breast cancer histology images.

In another study by Kwok, [3], four DCNN architectures i.e. VGG19, InceptionV3, InceptionV4, and InceptionResnetV2 were employed for the classification of H&E stained histological breast cancer images for both multi-class and binary classification. In that study, 5600 patches with the size of 1495×1495 and a stride of 99 pixels are extracted from the images. In Kwok's study, InceptionResnetV2 achieved the highest accuracy of 79% for multi-class and 91% for binary classification.

Another research study conducted by Sarmiento et al. [4] proposed a machine learning approach using feature vectors extracted from different characteristics such as shape, color, texture, and nuclei from each image. A Support Vector Machine (SVM) classifier with a quadratic kernel with 10-fold cross-validation was used to classify images but only achieved an overall accuracy of 79.2%.

Nawaz et al. [5] employed a fine-tuned AlexNet architecture for automatic breast cancer classification. The patches with the size of 512×512 pixels from training images were extracted and achieved an overall accuracy of 75.73% for the patch-wise dataset and 81.25% for the image-wise dataset.

A project similar to this project has been made by Sara Hosseinzadeh Kassani and her team. [6] They used transfer learning with the Xception model, and they achieved an mAP of 92.5% on their test data set.

3
Feasibility

3.1 Legal Feasibility

The licenses of Python [7], Keras [8], and Tensorflow [9], which are planned to be used within the scope of the project, have been reviewed and approved for use within the scope of the project.

3.2 Economic Feasibility

The project does not have an economical expense.

3.3 Technical Feasibility

3.3.1 Hardware Feasibility

The Keras [8] library, which is planned to be used within the scope of the project, does not need high GPU power for learning operations. Personal computers were used in project development.

System features of the computer where the project will be performed.
Processor : Intel(R) Core(TM) i7-6700HQ CPU @ 2.60 GHz
RAM : 16 GB
Display card : NVIDIA GeForce GTX 960M

3.3.2 Software Feasibility

The project will be implemented on Windows 10. The project will be written in Python [7] language (Python 3.7), and Tensorflow [9] (GPU), Keras [8] libraries will be used. Keras is an open-source library for artificial neural networks written in Python.

3.3.3 Workforce and Time Schedule

Figure 3.1 shows which tasks will be performed on which dates in the Gantt Diagram Table. In Figure 3.2, these tasks are expressed as a Gantt Chart.

Project	T	01/10/2020	01/01/2021	67 days	%
Determine of the project	T	01/10/2020	15/10/2020	11 days	%
Preliminary research	T	15/10/2020	28/10/2020	10 days	%
Meeting	M	27/10/2020	27/10/2020	-	%
Dataset analysis	T	28/10/2020	05/11/2020	7 days	%
Coding	T	05/11/2020	07/12/2020	23 days	%
Test process	T	07/12/2020	15/12/2020	7 days	%
Analysis of BACH dataset	T	15/12/2020	01/01/2021	14 days	%
Reporting	T	01/10/2020	11/01/2021	73 days	%

Figure 3.1 Gantt Diagram Table

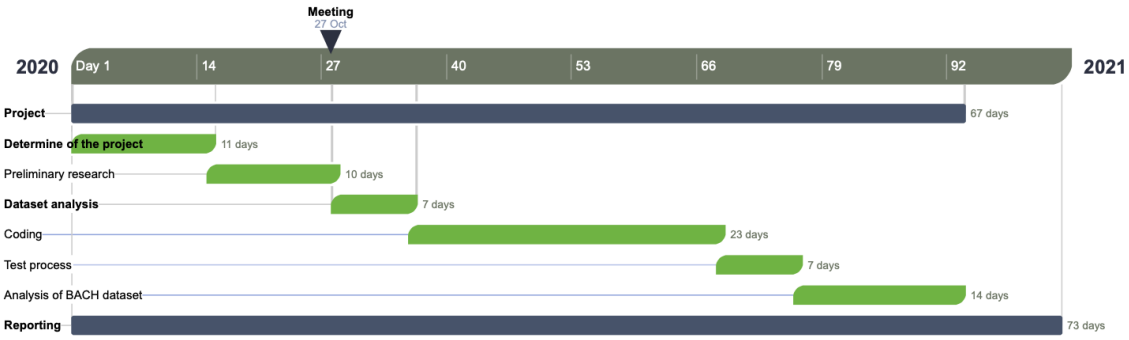


Figure 3.2 Gantt Chart

4
System Analysis

During the project, various deep learning models and methods are used, and it is planned to find the best setup with high success rates for the BACH and Bioimaging datasets.

4.1 Transfer Learning

Transfer or inductive learning is a supervised learning technique that reuses a previously trained model on a new network tasked for a different but similar problem. The goal is to create a framework that is at least better than a naïve model, so you can be assured that some new feature learning takes place. Typically deeper layers are reused as these tend to be more general, whereas the top layers tend to be more finely tuned to a particular problem. The new model is trained on the new data set as usual. The advantage is that the model tends to convergence much faster, and thus fewer data and compute time is required.

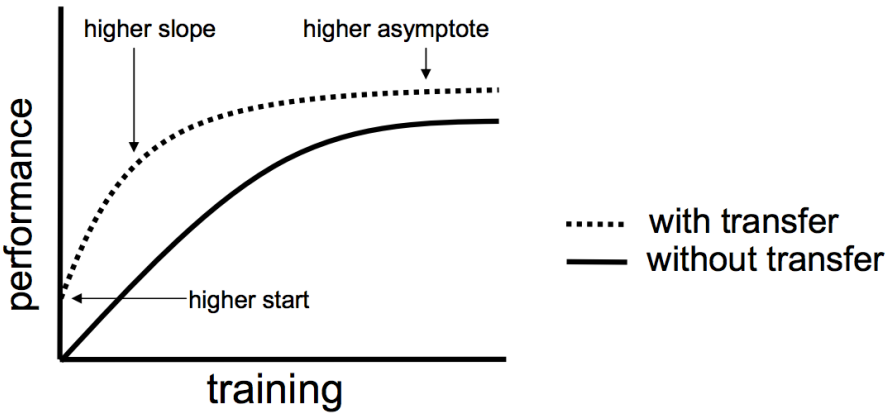


Figure 4.1 Transfer Learning Performance [10]

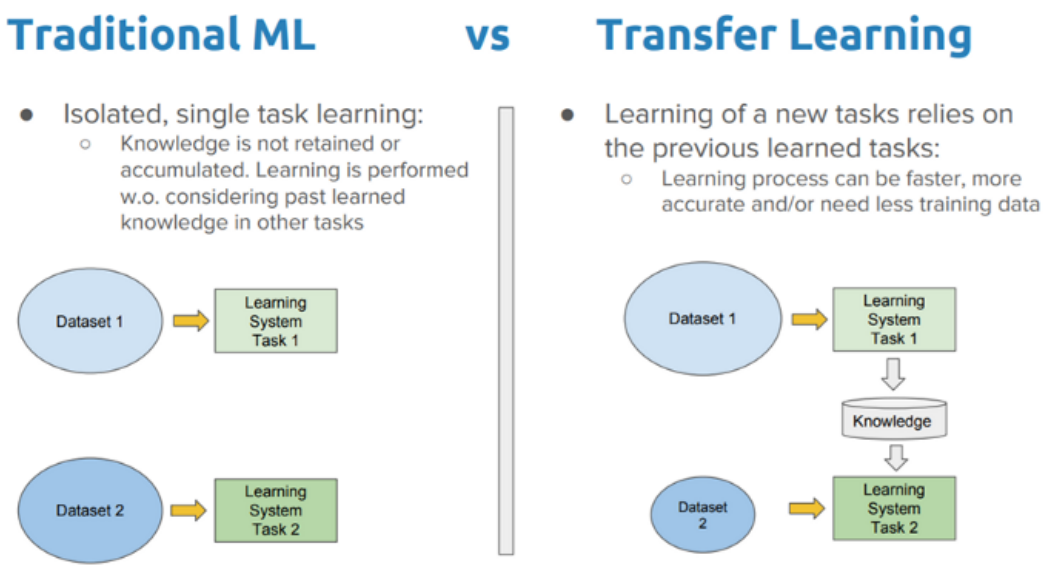


Figure 4.2 Traditional vs Transfer Learning [11]

4.2 Tensorflow

Tensorflow is a library developed by Google for machine learning. Google uses this library in most technologies. As an example, we use this library when searching by voice or image in the search engine. There are many versions of the library. 2.3.0 version is used in the project.

4.3 Keras

Keras is a deep learning framework for Python that provides a convenient way to define and train almost any type of deep learning model. Keras is a high-level neural network API written in Python, which can run on Tensorflow, Theano, and CNTK. It has been developed for fast experiments. Keras is a learning framework for Python used to train the model with deep learning and transfer learning. Keras models are developed for rapid experiments.

Keras has the following features :

- Allows for easy and fast prototyping.
- Run seamlessly on CPU and GPU.
- Supports both convolutional networks(for computer vision) and recurrent networks(for sequence and time-series), as well as the combination of two.

- It supports arbitrary network architectures: multi-input or multi-output models, layer sharing, model sharing, and so on. This means Keras is appropriate for building deep learning models, from generative adversarial networks to a neural Turing machine.

4.4 PyQt5 Designer

PyQt5 comes with a simple designer application for us to design our application. After designing the front face with PyQt5 Designer, the front face is converted from the command line to the python code, and the necessary codes for the button and other interface elements are coding.

5
System Design

5.1 Dataset Analysis

The histopathology images used in the application within the scope of the project were obtained from the H&E (Hematoxylin and Eosin) stained breast cancer reflections and the BACH(Breast Cancer Histology) dataset, which were presented for the clinical breast cancer classification competition at the ICIAR 2018 grand challenge [12]. The data set has been marked by two pathologists, and the spatial pixel resolution of the histopathological images used is 0.42 µm x 0.42 µm. The frame size of each image is 2084 × 1536 pixels. The data set has a 4-class structure normal, benign, in situ carcinoma, and invasive carcinoma. The BACH data set is an expanded version of the pictures in the Bioimaging 2015 competition [13].

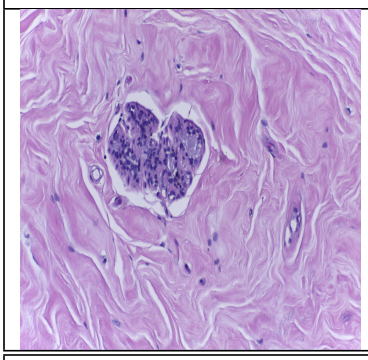
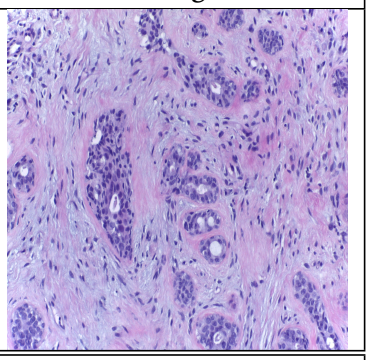
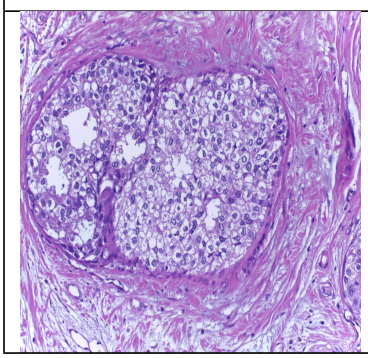
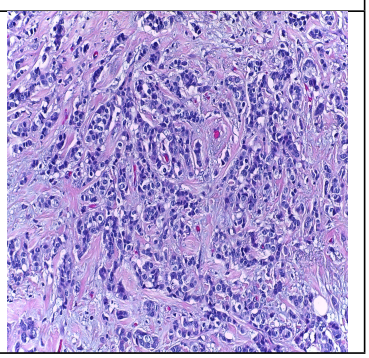
Normal	Benign
	
In-Situ	**Invasive**
	

Table 5.1 BACH Dataset Classes

5.2 Stain Normalization

One of the major difficulties in histopathology image analysis is appearance variability. Stain Normalization performs staining unmixing (separation of the hematoxylin and eosin stains), and appearance normalization [14].

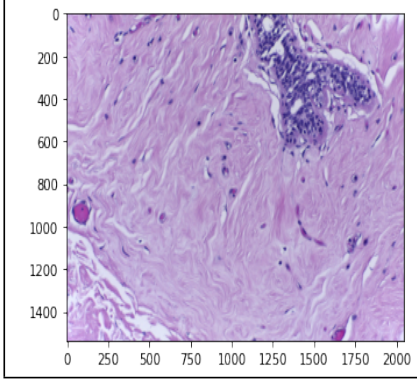
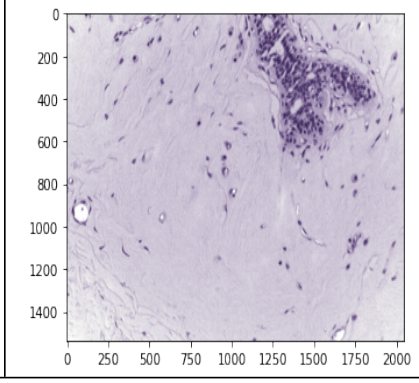
Original Image	Stain Normalization Image
	

Table 5.2 Stain Normalization

5.3 Gray-scale Conversion

Gray-scale is the collection or the range of monochromic (gray) shades, ranging from pure white on the lightest end to pure black on the opposite end. Gray-scale only contains luminance (brightness) information and no color information; that is why maximum luminance is white, and zero luminance is black; everything in between is gray. If you remove the color information, you are left with a gray-scale, resulting in a black and white image.

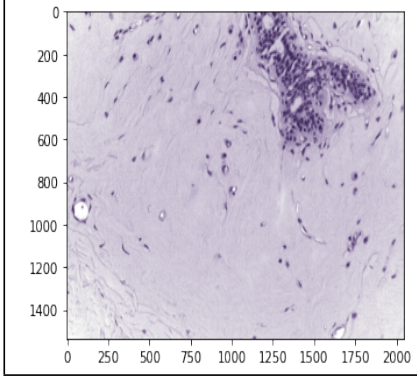
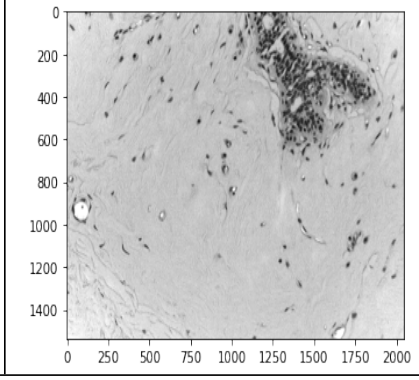
Stain Normalization Image	Gray Scale Image
	

Table 5.3 Gray Scale Conversion

5.4 Thresholding

For every pixel, the same threshold value is applied. If the pixel value is smaller than the threshold, it is set to 0; otherwise, it is set to a maximum value. The threshold value is used to classify the pixel values.

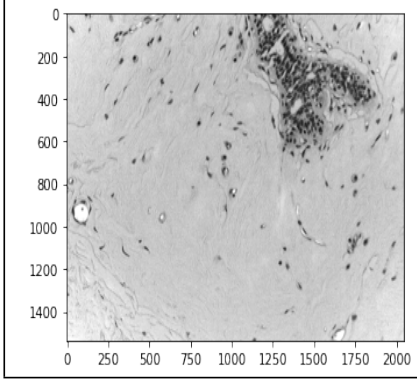
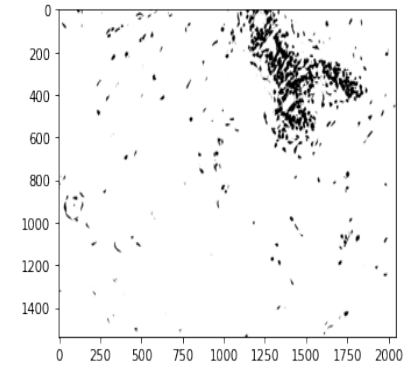
Gray Scale Image	Black and White Image
	

Table 5.4 Thresholding

5.5 Patch Extraction

We converted the stain normalization image to a black and white image using a threshold value for each image. Then we divided the image into patches. When we divide the image into patches, we use the step size as 8. To choose patches, we used the density of cells/background in a patch. If the rate of cells above the threshold value, we added the patch to the patch list. If the patch list is above 20, we got the first 20 patches from the sequential patches.

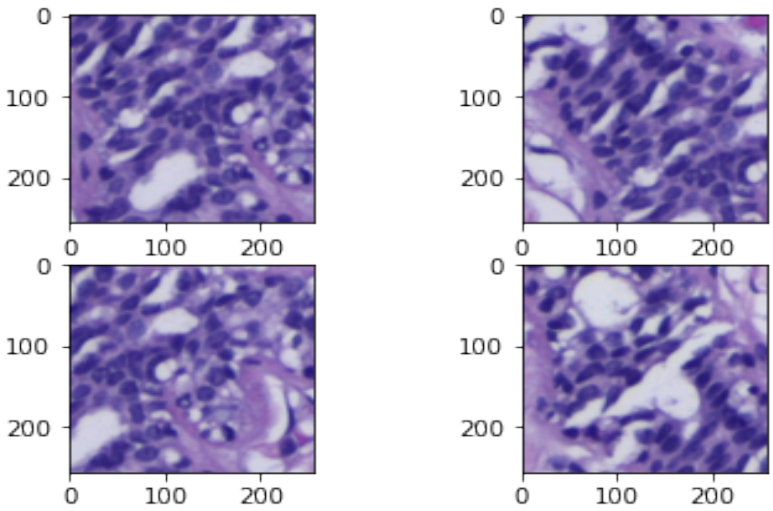


Table 5.5 Patched Images

5.6 Patched Images

We use patched images to evaluate model accuracy. We used 3 types patched. Our patch sizes are 256, 128 and 75. Also our step size is 8.

		Patch Sizes			
Dataset	**Classes**	**Original Set**	**256x256**	**128x128**	**75x75**
BACH 2018 (Train Data)	Benign	100	815	1452	1863
	In-Situ	100	777	1552	1985
	Invasive	100	422	949	1635
	Normal	100	1267	1834	1994
Total:		400	3281	5787	7477
Bioimaging 2015 (Test Data)	Benign	69	596	952	1243
	In-Situ	63	577	951	1245
	Invasive	62	160	545	998
	Normal	55	571	954	1094
Total:		249	1904	3402	4580

Table 5.6 Numerical information about the datasets

5.7 Data Augmentation

Increasing the data by exposure to various distortion effects improve the performance, especially in small data sets. In this way, the model is provided to learn about different conditions. The methods used in the interface are listed below;

- Rotation Range
- Height-Width Shift Range
- Zoom-Shear Range
- Horizontal Flip

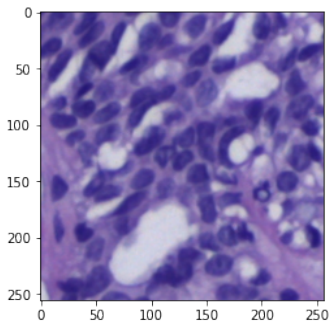
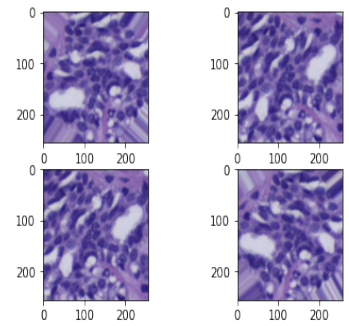
Original Image	Data Augmentation Images
	

Table 5.7 Data Augmentation

5.8 Bottleneck Features

The basic technique to get transfer learning working is to get a pre-trained model (with the weights loaded) and remove final fully-connected layers from that model. Then used the remaining portion of the model as a feature extractor for a small dataset. These extracted features are called "Bottleneck Features" (i.e., the last activation maps before the original model's fully-connected layers).
We then train a small fully-connected network on those extracted bottleneck features to get the classes we need as outputs for our problem.

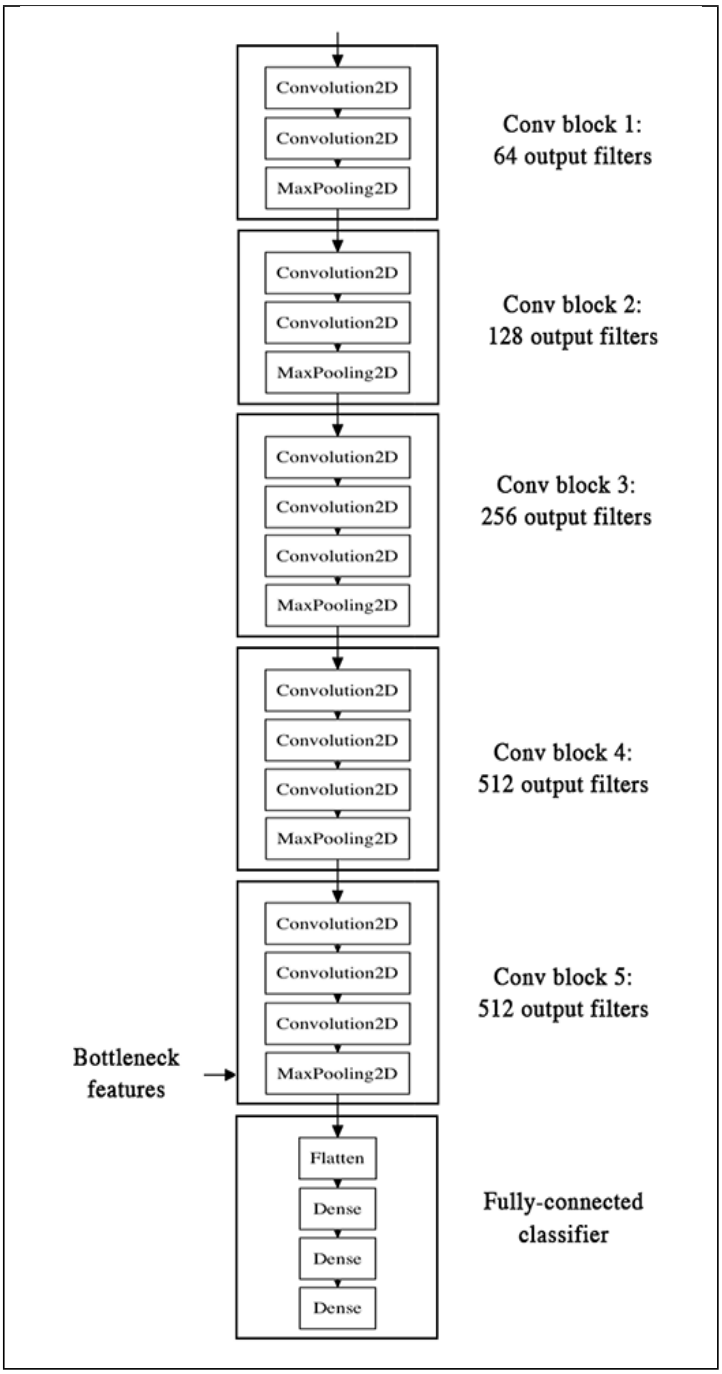


Table 5.8 Bottleneck Feature Extraction in the VGG16 Model

5.9 K-Fold Cross Validation

Randomly split an entire dataset into k-folds. For each k-fold in the dataset, we built a model on k–1 folds of the dataset. Then, we tested the model to check the effectiveness of the kth fold. Recorded the accuracy seen on each of the predictions. Repeated this until each of the k-folds has served as the test set. The average of k recorded accuracies is called the cross-validation accuracy and will serve as performance metric for the model.

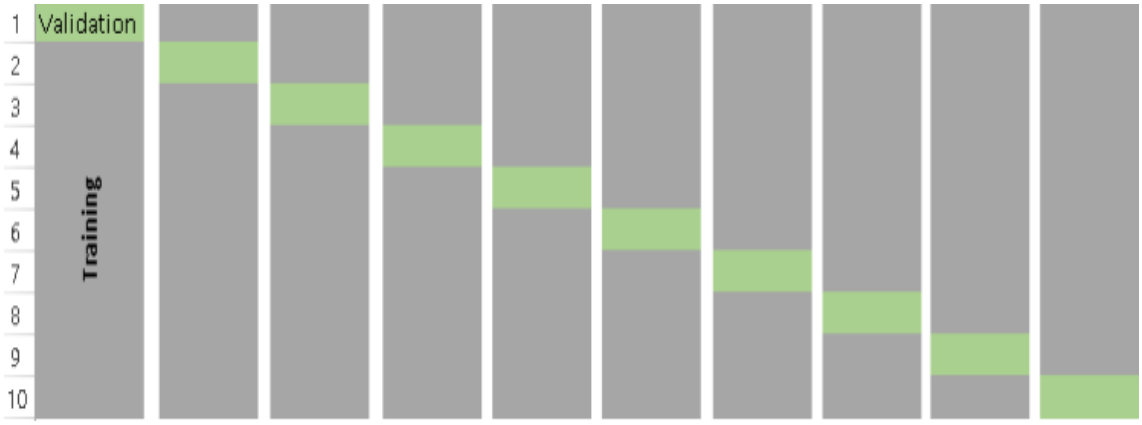


Table 5.9 Visualization of a K-Fold Validation

5.10 Keras Models

- DenseNet
  DenseNet tries to eliminate color transition using a different approach. Instead of using shortcut links, all previous feature maps become the input of the next layer. There are 121, 169, and 201 layered versions. As the number of layers increases, the number of parameters increases, and the duration of training increases. The default input size for this model is 224 × 224 [15].

- NASNetLarge
  Neural Architecture Search Network NASNet is developed by Google. The developers propose to look for an architectural building block in a small dataset and then transfer the block to a larger dataset. Finally, the NASNet model achieves the most advanced results with a smaller model size and lower complexity (FLOPS). NasNetMobile for mobile applications and NasNetLarge for advanced applications have been developed to improve optimization [16].

- Xception
  It is developed by Francois Chollet. Xception is a similar extension of the inception architecture. The image input size of the network is 224 × 224 [17].

- VGG16

  It is a simple network model. The most important difference from previous models is the use of convolutional additions with 2 or 3. Softmax performance of 1000 classes is calculated on the output of two Fully Connected layers. Approximately 138 million parameters have been calculated. As with other models, the number of channels increases, while the height and width dimensions of the matrices from input to output decrease. The image input size of the network is 224 × 224 [18].

- VGG19

  VGG-19 is a 19 fold deep curved neural network. You can download a pre-prepared version of the network trained on more than a million images from the ImageNet database. The network has learned rich feature representations for a wide variety of images. The image input size of the network is 224 × 224 [18].

- InceptionV3

  InceptionV3 tried to find a solution to the correct core size selection problem caused by diversity in datasets. As a solution to this problem, it has adopted combining outputs with multiple different size liters in the same layer. The default input size for this model is 299 × 299 [19].

- InceptionResNetV2

  They are very well-performing architectures with relatively low computing cost for both Initial and Residual networks. Inception-ResNet combines two architectures to improve performance further. The default input size for this model is 299 × 299 [20].

- ResNet

  ResNet 50, 101, and 152 are deep curved neural network models. Networks learned feature-rich presentations for a wide variety of images. The 2nd version of the networks was created as 50V2, 101V2, and 152V2. The image input size of the network is 224 × 224 [17].

5.11 Activation

The activation function is an operational "gate" between the input that feeds the current neuron and its output to the next layer. They decide whether the neuron is activated or not.

- Softmax
  Some vector components can be negative or greater than one; After applying softmax, each component will have values in the range (0,1) [21].

5.12 Optimizer

An optimizer is one of the two arguments required for compiling a Keras model.

- SGD
  Stochastic Gradient Landing is one of the simplest optimization algorithms. All parameters use only one static learning rate during the training phase. As optimizers approach the optimal value, they reduce gradients [22].

5.13 Loss Function

When you try your algorithm to try and develop your model, your lost function will tell you if you have reached anywhere. You should use the appropriate loss function to reach the highest result according to the dataset you use.

- Categorical Cross-entropy
  Categorical cross-entropy is used in multi-classification. In this classification structure, classes are represented by 0 and 1 [23].

6
Application

Images in the BACH and Bioimaging dataset are divided into small pieces as patches. Before we split images into patches, we apply stain normalization and threshold to images to choose the correct patches. The chosen patched images have been reproduced with data augmentation—created bottlenecks from these reproduced images using pre-trained models. Applied 2 methods to evaluate accuracy. Used k-fold cross-validation to get BACH dataset accuracy. The pictures are also separated as train data set where the training dataset is the BACH dataset, and the test dataset is the Bioimaging dataset to get Bioimaging dataset accuracy. The inference will be made with the obtained accuracy rates.

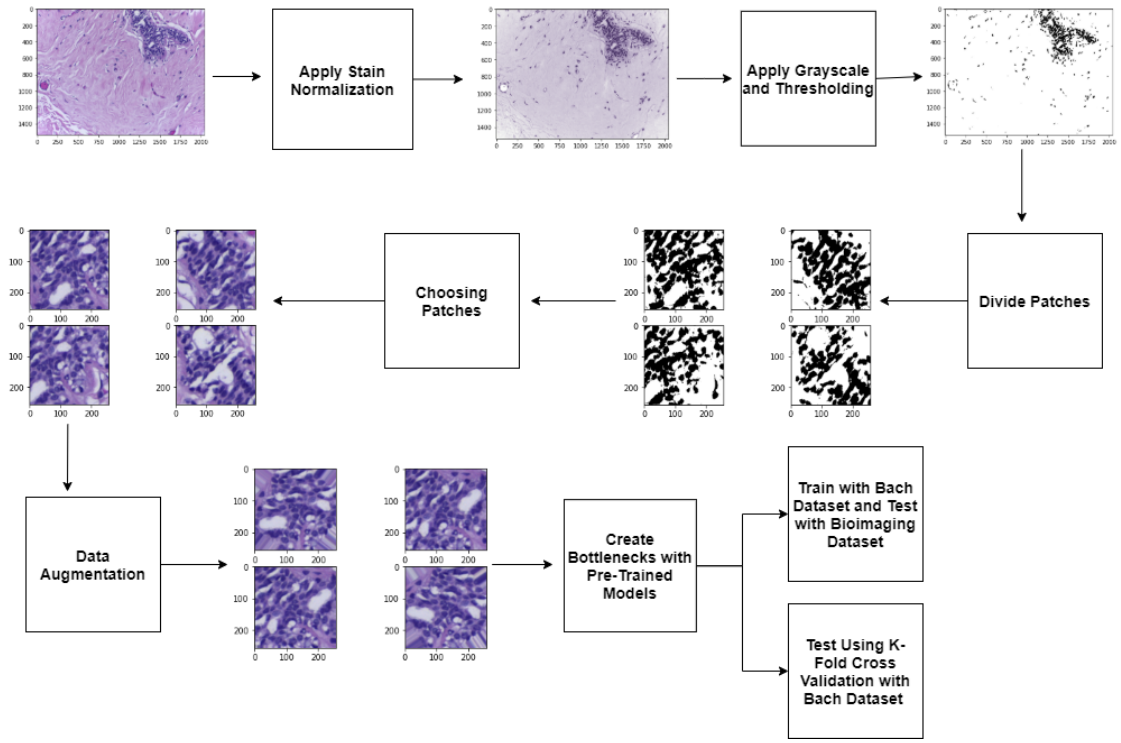


Figure 6.1 Sequential Steps in the Project

6.1 Interface

The interface consists of two parts as testing and prediction. You can test BACH and Bioimaging dataset. You can change pre-trained models to test. Also you can change patch size to select image size to test. You can also edit epoch size to train model. You can try BACH and Bioimaging dataset by entering variables. When the testing is over, you can review the details by outputs and confusion matrix on the screen. You can experiment with different test data by loading the model and weight you recorded in the test section. The main page of the interface is in figure 6.2 below.

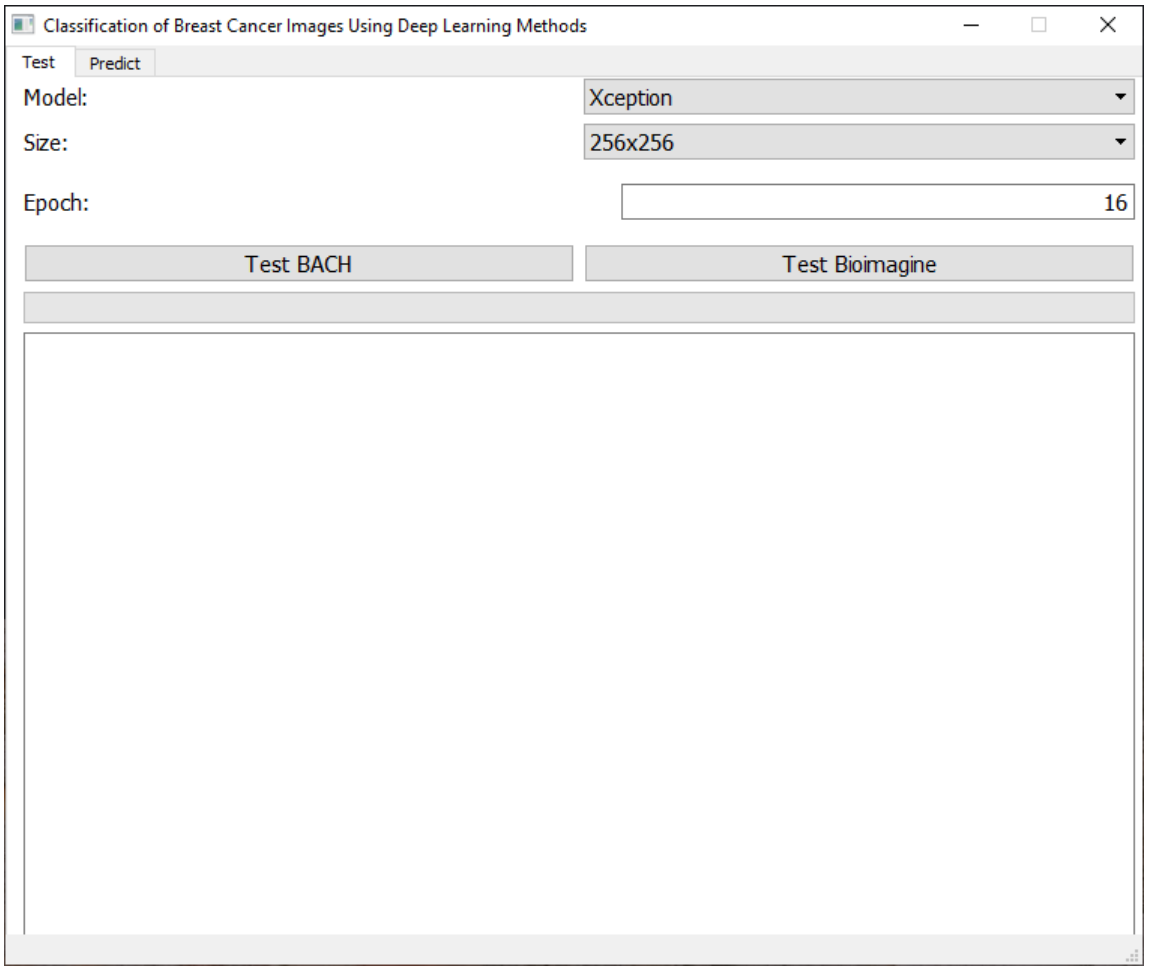


Figure 6.2 Application of Interface

6.2 Model Section

In this section, the necessary pre-trained model is taken from the user before compiling the model. The model is created by selecting one of the models defined on the interface. The accepted model is used to choose recorded bottleneck files. The sequential model is trained and tested with this bottleneck files.

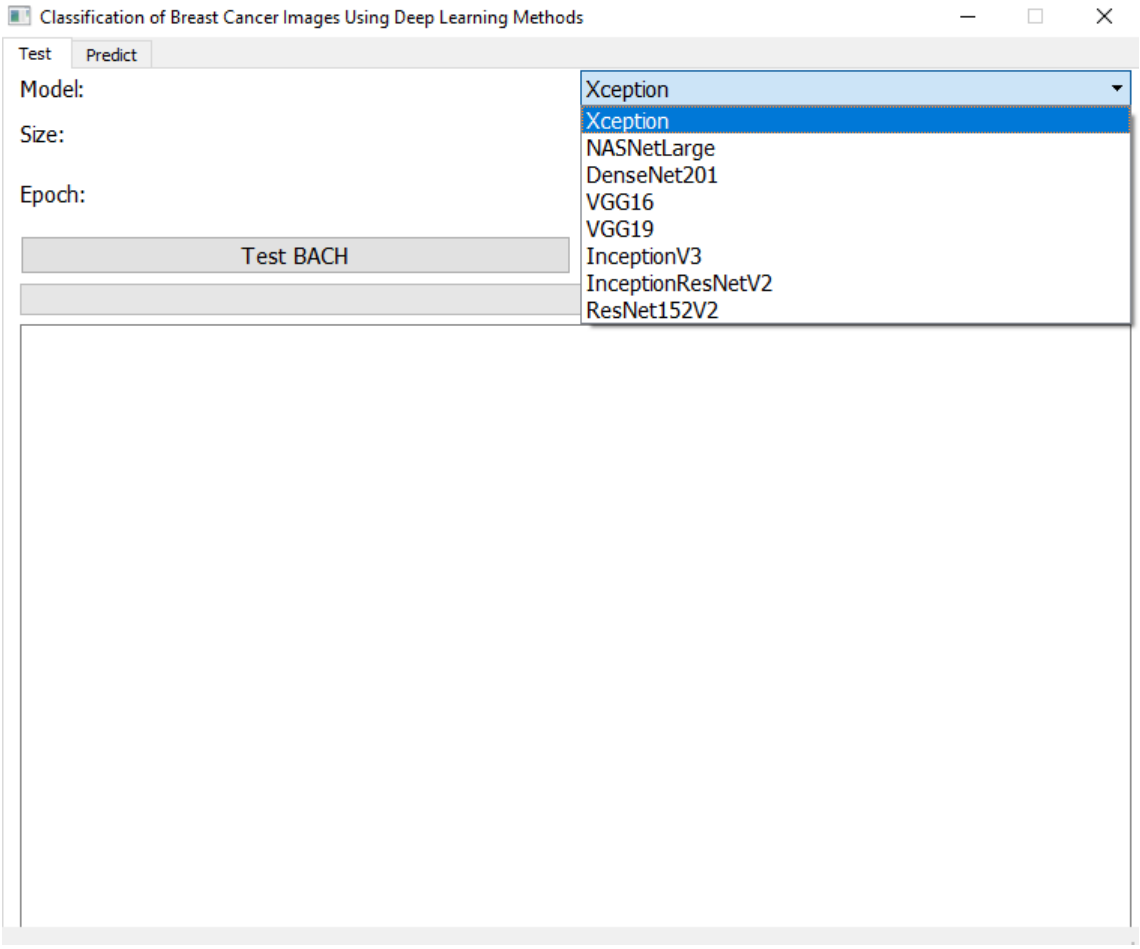


Figure 6.3 Choosing Model

6.3 Batch Size Section

In this section, the image size is taken from the user before compiling the model. The accepted batch size is used to choose recorded bottleneck files. The sequential model is trained and tested with this bottleneck files.

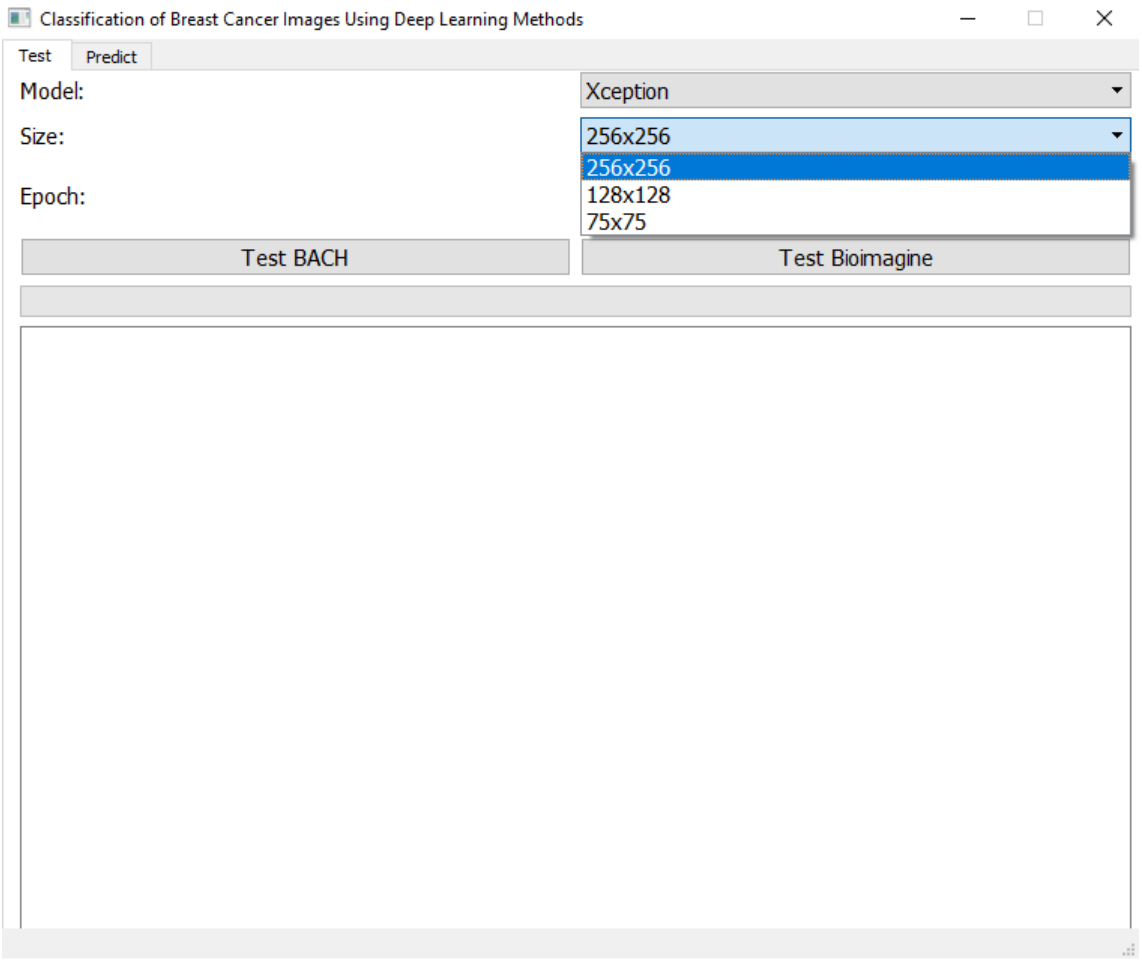


Figure 6.4 Choosing Batch Size

6.4 Test Bach Section

In this section, we were testing BACH dataset with taken variables from the user in the model section, batch size section, and epoch section. To test the BACH dataset, we used the k-fold cross-validation method. Also, we used bottleneck files, which before saved to disk for taken model and batch size. When the testing is over, you can review the details by outputs and confusion matrix on the screen.

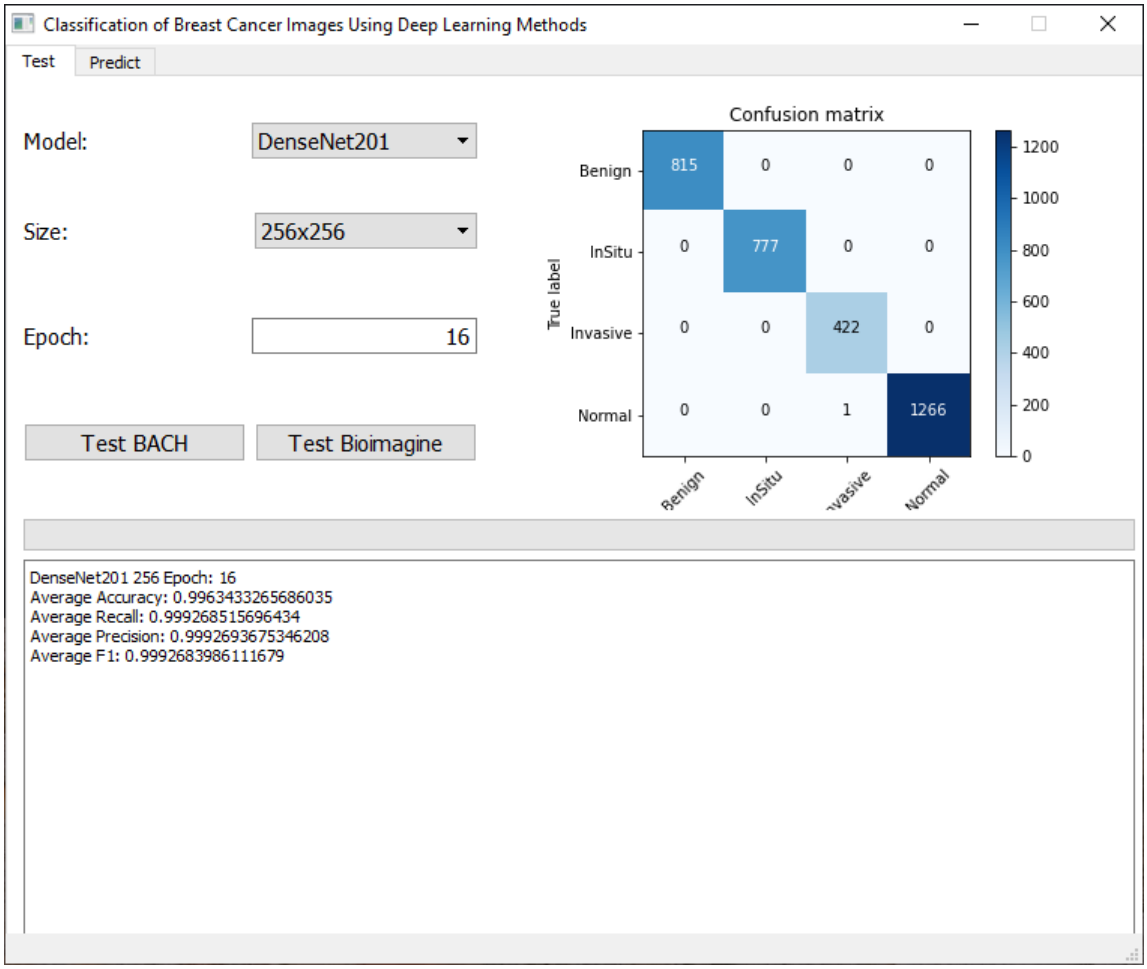


Figure 6.5 Outputs of Interface

6.5 Test Bioimaging Section

In this section, we are testing the Bioimaging dataset with taken variables from the user in the model section, batch size section, and epoch section. To test the Bioimaging dataset, we used to train and test the method where the BACH dataset is the training dataset, and the Bioimaging dataset is the test dataset. Also, we used bottleneck files, which before saved to disk for taken model and batch size. When the testing is over, you can review the details by outputs and confusion matrix on the screen.

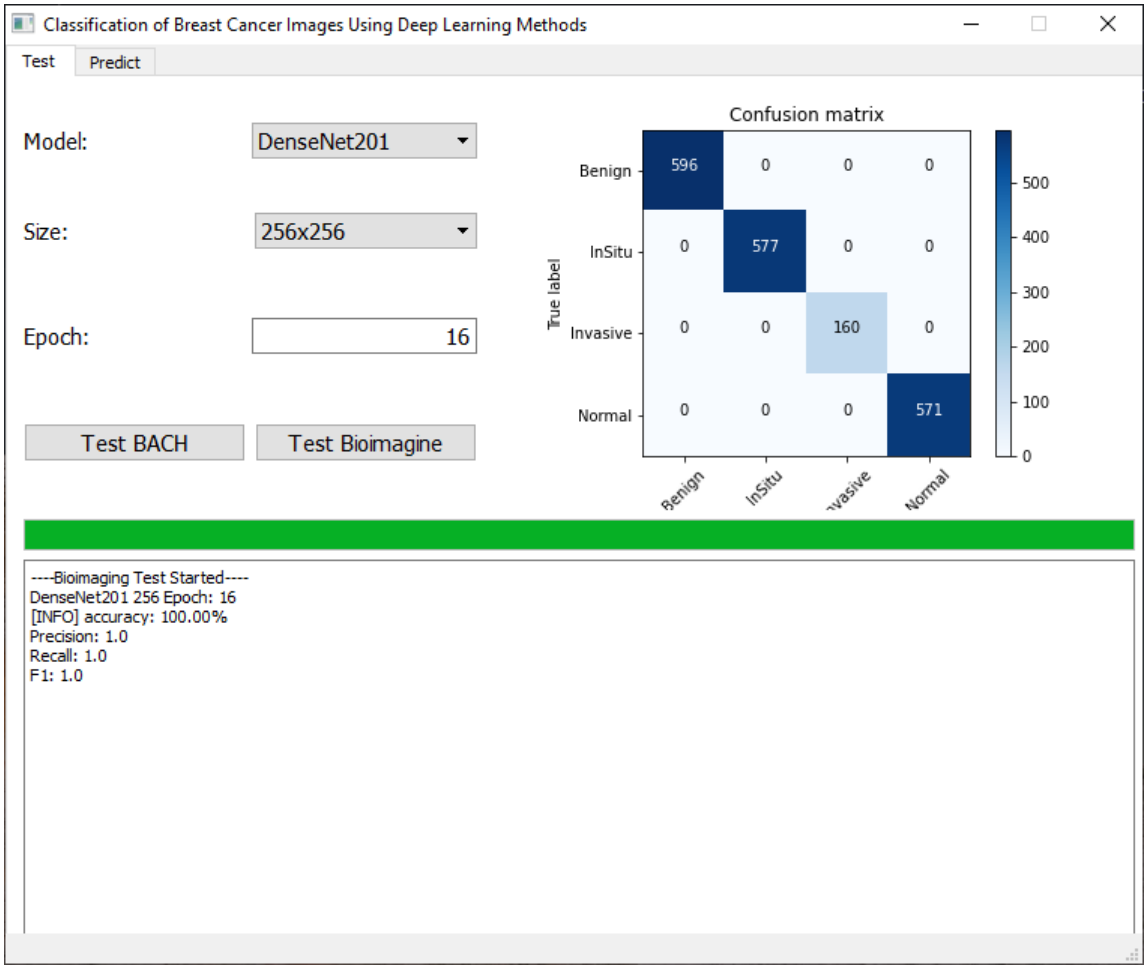


Figure 6.6 Outputs of Interface

6.6 Predict Section

In this section, you can predict images with taken variables from the user in the model section, batch size section, and epoch section. To predict images, we use model weights, which before saved to disk for taken model and batch size. When the process is started, chosen one image randomly for each class; after the process, you can review chosen images, true class names, and predicted class names.

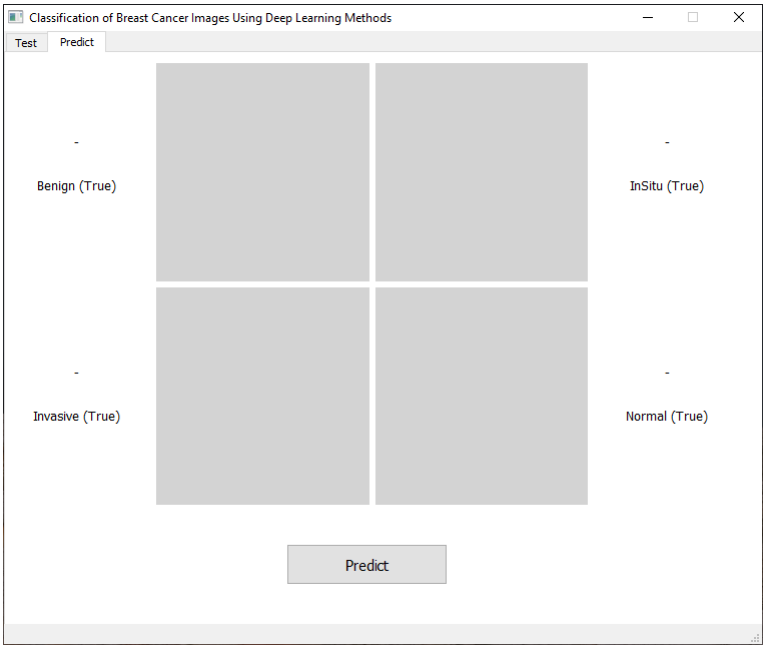


Figure 6.7 Predict Home Screen

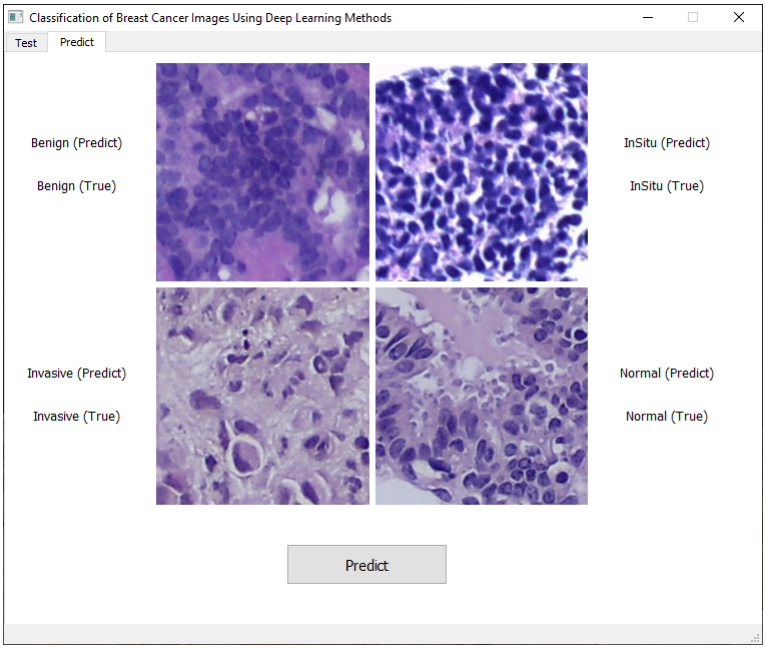


Figure 6.8 Predict Results Screen

6.7 Predict Single Section

In this section, a single picture is predicted. By clicking the 'Add Image' button, we select the image we want from the file explorer, and we provide the prediction by pressing the 'Predict' button.

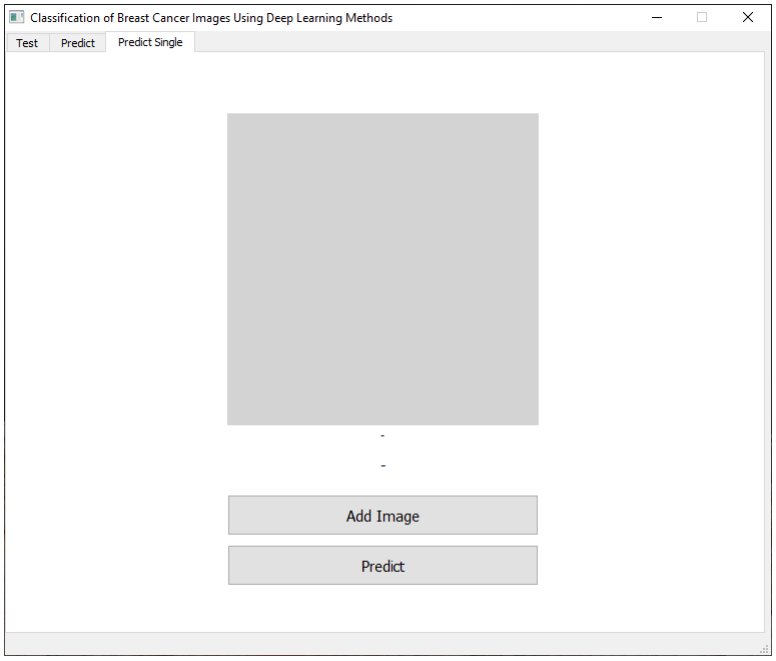


Figure 6.9 Predict Single Home Screen

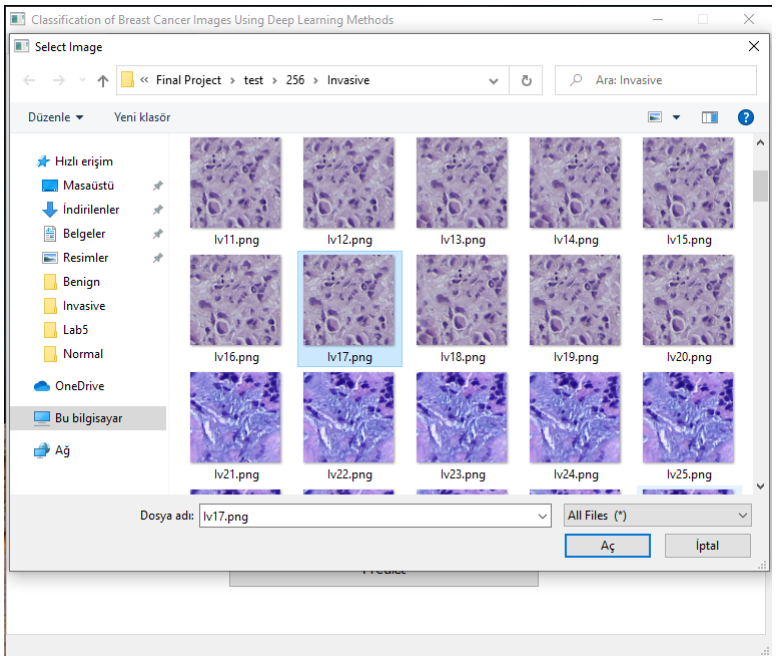


Figure 6.10 Predict Single Image Selection Screen

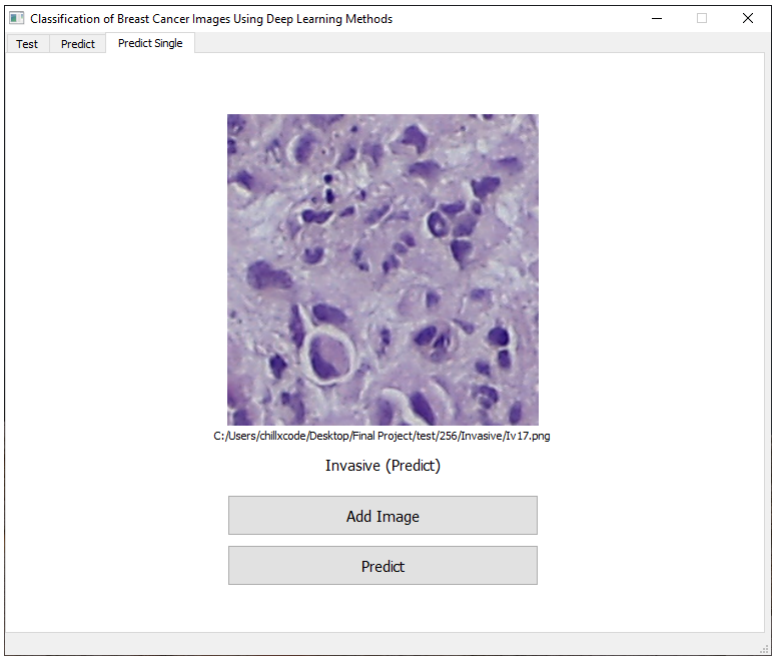


Figure 6.11 Predict Single Showing Predition Screen

7
Performance Analysis

Batch size tests were performed on the BACH and Bioimaging dataset with DenseNet201, NASNetLarge, Xception, VGG16, VGG19, InceptionV3, InceptionResNetV2, and ResNet152V2 Keras networks. We have three types of patch size. 256, 128, 75.

	Bach Dataset				**BioImaging Dataset**			
Methods	**Acc.**	**Pre.**	**Rec.**	**F1**	**Acc**	**Pre.**	**Rec.**	**F1**
DenseNet201	0.996	0.999	0.999	0.999	1.000	1.000	1.000	1.000
NASNetLarge	0.992	0.998	0.998	0.998	1.000	1.000	1.000	1.000
Xception	0.993	0.998	0.998	0.998	1.000	1.000	1.000	1.000
VGG16	0.604	0.604	0.608	0.535	0.943	0.947	0.943	0.943
VGG19	0.502	0.551	0.500	0.435	0.960	0.961	0.960	0.959
InceptionV3	0.990	0.998	0.998	0.998	1.000	1.000	1.000	1.000
InceptionResNetV2	0.992	0.998	0.998	0.998	1.000	1.000	1.000	1.000
ResNet152V2	0.993	0.998	0.998	0.998	1.000	1.000	1.000	1.000

Table 7.1 Accuracy rates when receiving 256x256 patch

	Bach Dataset				**BioImaging Dataset**			
Methods	**Acc**	**Pre.**	**Rec.**	**F1**	**Acc.**	**Pre.**	**Rec.**	**F1**
DenseNet201	0.957	0.989	0.985	0.989	0.999	0.999	0.999	0.999
NASNetLarge	0.784	0.892	0.881	0.881	0.942	0.944	0.942	0.942
Xception	0.871	0.922	0.921	0.921	0.965	0.966	0.965	0.965
VGG16	0.711	0.739	0.735	0.731	0.773	0.787	0.773	0.767
VGG19	0.681	0.741	0.712	0.707	0.721	0.744	0.721	0.713
InceptionV3	0.844	0.947	0.947	0.947	0.985	0.985	0.985	0.985
InceptionResNetV2	0.861	0.929	0.928	0.928	0.963	0.963	0.963	0.963
ResNet152V2	0.914	0.981	0.981	0.981	0.999	0.999	0.999	0.999

Table 7.2 Accuracy rates when receiving 128x128 patch

	Bach Dataset				BioImaging Dataset			
Methods	**Acc.**	**Pre.**	**Rec.**	**F1**	**Acc.**	**Pre.**	**Rec.**	**F1**
DenseNet201	0.719	0.822	0.819	0.819	0.873	0.875	0.873	0.873
NASNetLarge	0.471	0.558	0.547	0.542	0.569	0.591	0.569	0.566
Xception	0.613	0.693	0.681	0.681	0.761	0.772	0.761	0.760
VGG16	0.488	0.525	0.505	0.499	0.572	0.584	0.572	0.566
VGG19	0.492	0.527	0.509	0.502	0.548	0.562	0.548	0.535
InceptionV3	0.545	0.625	0.622	0.620	0.639	0.641	0.639	0.638
InceptionResNetV2	0.533	0.600	0.577	0.576	0.648	0.660	0.648	0.646
ResNet152V2	0.604	0.838	0.835	0.834	0.901	0.906	0.901	0.901

Table 7.3 Accuracy rates when receiving 75x75 patch

References

[1] N. C. Institute. (2020). "National cancer institute website," [Online]. Available: https://www.cancer.gov (visited on 11/30/2020).

[2] V. I. A. Rakhlin A. Shvets and A. Kalinin, "Deep convolutional neural networks for breast cancer histology image analysis," *International neural networks for breast cancer histology image analysis*, pp. 737–744, 2018.

[3] S. Kwok, "Multiclass classification of breast cancer in whole-slide images," *International Conference Image Analysis and Recognition*, pp. 931–940, 2018.

[4] A. Sarmiento and I. Fondon, "Automatic breast cancer grading of histological images based on colour and texture descriptors," *International Conference Image Analysis and Recognition*, pp. 887–894, 2018.

[5] S. A. W. Nawaz and H. Khan, "Classification of breast cancer histology images using alexnet," *International Conference Image Analysis and Recognition*, pp. 869–876, 2018.

[6] M. W. S. Kassani P. Kassani and R. Deters, "Breast cancer diagnosis with transfer learning and global pooling," *ICTC 2019*, pp. 519–524, 2019.

[7] P. S. Foundation. (2001). "Python programming language," [Online]. Available: https://www.python.org (visited on 11/30/2020).

[8] K. Team. (2020). "Python deep learning library," [Online]. Available: https://keras.io (visited on 11/30/2020).

[9] T. Team. (2020). "Tensorflow: Large-scale machine learning on heterogeneous systems," [Online]. Available: https://www.tensorflow.org (visited on 11/30/2020).

[10] M. L. Mastery. (2020). "A gentle introduction to transfer learning for deep learning," [Online]. Available: https://machinelearningmastery.com/transfer-learning-for-deep-learning/ (visited on 11/30/2020).

[11] T. D. Science. (2020). "A comprehensive hands-on guide to transfer learning with real-world applications in deep learning," [Online]. Available: https://towardsdatascience.com/a-comprehensive-hands-on-guide-to-transfer-learning-with-real-world-applications-in-deep-learning-212bf3b2f27a (visited on 11/30/2020).

[12] grandchallenge.org. (2021). "Iciar 2018 grand challenge," [Online]. Available: https://iciar2018-challenge.grand-challenge.org (visited on 01/10/2021).

[13] bioimaging2015.com. (2021). "Bioimaging 2015 the pre-clinical challenge," [Online]. Available: http://www.bioimaging2015.ineb.up.pt/challenge_overview.html# (visited on 01/10/2021).

[14] M. Macenko. (2021). “Normalize staining,” [Online]. Available: https://github.com/wanghao14/Stain_Normalization (visited on 01/10/2021).

[15] A. L. G. Huang and L. van der Maaten. (2021). “Densely connected convolutional networks,” [Online]. Available: https://arxiv.org/pdf/1608.06993.pdf (visited on 01/10/2021).

[16] J. S. Barret Zoph Vijay Vasudevan and Q. V. Le, “Learning transferable architectures for scalable image recognition,” 2018.

[17] S. R. Kaiming He Xiangyu Zhang and J. Sun, “Deep residual learning for image recognition,” 2015.

[18] K. Simonyan and A. Zisserman. (2021). “Very deep convolutional networks for large-scale image recognition,” [Online]. Available: https://arxiv.org/pdf/1409.1556.pdf (visited on 01/10/2021).

[19] Google. (2021). “Going deeper with convolutions,” [Online]. Available: https://arxiv.org/pdf/1409.4842v1.pdf (visited on 01/10/2021).

[20] S. I. Christian Szegedy and V. Vanhoucke, “Inception-v4, inception-resnet and the impact of residual connections on learning,” 2016.

[21] K. Team-A. (2020). “Layer activation functions,” [Online]. Available: https://keras.io/api/layers/activations/ (visited on 11/30/2020).

[22] K. Team-B. (2020). “Optimizers,” [Online]. Available: https://keras.io/api/optimizers/ (visited on 11/30/2020).

[23] K. Team-C. (2020). “Losses,” [Online]. Available: https://keras.io/api/losses/ (visited on 11/30/2020).

Curriculum Vitae

FIRST MEMBER

Name-Surname: Emre ÇELİK
Birthdate and Place of Birth: 02.05.1997, İzmir
E-mail: emrecelikk97@gmail.com
Phone: 0539 277 26 12
Practical Training: DSM Grup Danışmanlık İletişim ve Satış Ticaret A.Ş. - Dolap Tech

SECOND MEMBER

Name-Surname: Kamil BAŞKUT
Birthdate and Place of Birth: 11.11.1997, Kastamonu
E-mail: kamil.baskut1@gmail.com
Phone: 0541 925 61 60
Practical Training: Etiya Bilgi Teknolojileri Yazılım Sanayi ve Ticaret A.Ş.

Project System Informations

System and Software: Windows Operation System, Python
Required RAM: 16GB
Required Disk: 80GB