Cancer Classification Detection Based on BreakHis Dataset

Group 2: Qianyi Xue, Kejia Liu, Longxin Wang, and Yuchen Ge
ST 7015 Summary
Dr. Mark Vogelsberger
July 2, 2022

I. DATASET

In women worldwide, breast cancer is the most common cancer and the second leading cause of cancer death (Boyle & Levin, 2008). Detecting and diagnosing breast cancer from histopathological images with high accuracy have been enabled by recent advancements in deep learning. To conduct such computer-aided breast cancer diagnosis, our group utilized the Breast Cancer Histopathological Image Classification (BreakHis) database (Spanhol et al., 2016), which is composed of 7909 microscopic images of breast tumor tissue with four magnification factors (40x, 100x, 200x, and 400x). The tumors in the BreakHis are divided into two groups: benign and malignant, in which there are four subtypes of breast cancers correspondingly. The four benign breast tumors are adenosis, fibroadenoma, phyllodes tumor, and tubular adenoma, while the four malignant are carcinoma, lobular carcinoma, mucinous carcinoma, and papillary carcinoma.

The first 70% of the 7909 images were selected as training images; the following 15% of the data were selected as the validation set, whereas the remaining data belonged to the testing set.

Magnification	Benign	Malignant	Total
40×	625	1370	1995
100×	644	1437	2081
200×	623	1390	2013
400×	588	1232	1820
Total	2480	5429	7909
# Patients	24	58	82

Table 1. Image Distribution by Magnification Factor and Class (Spanhol et al., 2016).

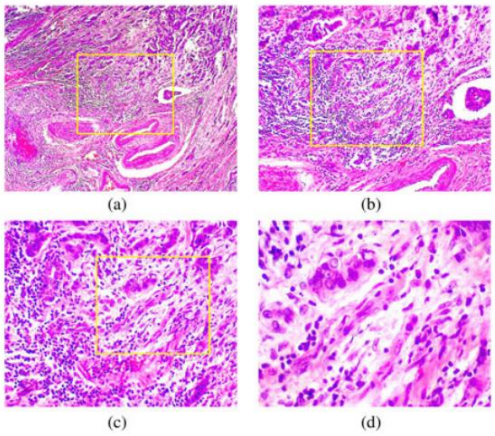

Figure1. Slide of breast malignant tumor (stained with HE) seen in different magnification factors: (a) 40×, (b) 100×, (c) 200×, and (d) 400× (Spanhol et al., 2016).

II. MODEL

Our overall goal is to construct a deep CNN for the BreakHis dataset such that the breast cancer images can be classified as accurately as possible. The overall progress is divided into 2 steps: **extracting features from the images and building the classifier.** We used 6 models to extract image features ($vgg16$, $vgg19$, $xception$, $resnet50$, $inception$, $and\ inception\_resnet$) and constructed classifiers based on machine learning methods (logistic regression, SVM, etc.) and fully connected layers for the feature vectors. We evaluated their performance mainly based on accuracy.

Vgg16 model by Simonyan et al. (2014) uses stacked small convolutional instead of large convolutional kernels. It has multiple nonlinear layers that can increase the network depth ensuring complex computation and less parameter computation. Inception model by Szegedy et al. (2015) uses some techniques to prune the network to get sparse weights or connections, which replaces the fully connected layer with a global mean pooling layer after the final convolution layer, thus further reducing the number of parameters. Resnet model by He, K et al. (2016) introduces a residual module enabling training when having deeper layers and at the same time avoid the problem of gradient explosion and gradient disappearance.

We constructed a concise architecture, taking a 5-layer dense layer for each model to facilitate rapid convergence, and the final output matrix of $[image\_num, 8]$(8 classes) is obtained as the extracted features.

```
def resnet50_model(load_weights = True):
    base_model = ResNet50(include_top=False, weights='imagenet', input_tensor=None,
input_shape=(image_height, image_width,3), pooling='avg')
    x = base_model.output
    x = Dense(1024, activation='relu')(x)
    x = Dense(256, activation='relu')(x)
    x = Dense(64, activation='relu')(x)
    x = Dense(16, activation='relu')(x)
    x = Dense(8, activation='softmax')(x)
    model = Model(inputs=base_model.input, outputs=x)
    model._name = 'resnet'
    return model
```

Code of resnet50

Regarding the training parameters, we set the upper limit of **epoch to 500**, the **batch size is 64** according to our computational ability, the initial value of learning rate to **0.00005**, our loss function to $"categorical\_crossentropy"$, and the strategy of early stopping and decreasing learning rate for training.

```
    model.compile(loss="categorical_crossentropy", optimizer=Adam(lr=0.00005),
metrics=['accuracy'])
    early_stopping = EarlyStopping(patience=10, verbose=2)
    model_checkpoint = ModelCheckpoint(model_name + "_combine" +".model",
save_best_only=True, verbose=2)
    reduce_lr = ReduceLROnPlateau(factor=0.1, patience=5, verbose=2) #min_lr=0.00001,
```

Key for training

We present the final accuracy of classification using fully connected layers.

	VGG16	VGG19	Xception	Resnet50	Inception	Inception_resnet
40x	0.807	0.786	0.722	0.702	0.725	0.837
100x	0.845	0.777	0.744	0.880	0.735	0.819
200x	0.855	0.784	0.774	0.872	0.784	0.865

400x	0.826	0.803	0.754	**0.883**	0.735	0.841

Table 2. Table of the accuracy of the classification results of different models with FC.

It can be seen from Table 2 that the optimal solution is reached on resnet50 and inception_resnet for images with different magnification (highlighted part), respectively. Averaging the values of different models at each magnification, the average classification accuracy increases with the increase of magnification. The global optimal accuracy can reach 88.3% (in the resnet model with magnification of 400x).

III. TRAINING TRICKS

Increasing the number of FC layers

Research by Krizhevsky et al. (2017) demonstrated that increasing the depth of the network is advantageous to extract more effective features and enhance the performance of the network. Therefore, keeping the width of the network constant, our group sought to improve the test accuracy via increasing the depth of the network. Our group increased the depth of the network by adding fully connected (FC) layers. Using approximately 70% (5481 out of 7909) of the images to train, we carried out an experiment by contrasting the test accuracies of three architectures: VGG16 with 5 FC layers, VGG16 with 6 FC layers, and VGG16 with 7 FC layers. The test accuracies are shown in Table 1.

magnification\model	VGG16 + 5 FC layers	VGG16 + 6 FC layers	VGG16 + 7 FC layers
40x	0.874	0.845	0.918
100x	0.888	0.921	0.870
200x	0.881	0.867	0.876
400x	0.907	0.918	0.907

Table 3. Test accuracies for three networks with VGG16 as the base model (batch size = 32).

Table 3 illustrates that adding FC layers is not completely beneficial for improving the test accuracy since the test accuracies for different magnifications do not blindly increase as the number of FC layers increases. More layers will help extract more features, which can be done up to a certain extent. In addition, adding layers increases the number of weights in the neuron network, ergo the complexity of the model. In this case, if the training set is not large enough, simply increasing the depth of the network will bring about more parameters, and thus the model risks overfitting and in turn resulting in a reduction in the accuracy of the test data. Hence, such a tradeoff should be taken into consideration when we try to improve the performance of the model.

Learning Rate Strategies

1. Learning Rate Decay

The learning rate is too large may cross the optimal value, and the learning rate is too small cannot converge for a long time. The basic idea of learning rate decay is that the learning rate decays gradually as the training progresses.

2. Cyclic Learning Rate Decay

When our group was investigating how to optimize the performance of the algorithm, we saw several ways to optimize the algorithm, and one of the methods of cyclic decay attracted my attention.

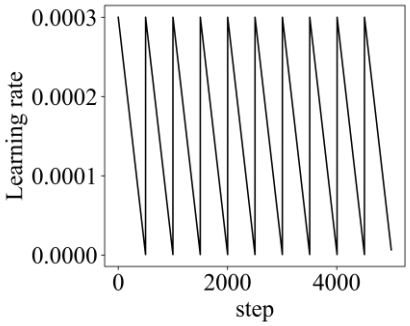

Figure 2. Simulation training using CLR.

3. Comparison of cyclical learning rate decay and general learning rate decay

This algorithm decreases the learning rate every 5 epochs, and the learning rate becomes 0.1 of the original for each reduction. The number of cycles to reduce learning rate is set to 5, which means that we do 5 cycles of reduced Lr for one training, and then stop for the next training. Next, we compare the accuracy of 40X, 100X, 200X, and 400X for 6 mods with and without cycles. The test accuracies without cyclical learning rates are shown in Table 4. The test accuracies with cyclical learning rates are shown in Table 5.

Magnification Model	VGG16	VGG19	Xception	Resnet50	Inception	Inception_resnet
40x	0.786	0.647	0.680	0.702	0.675	0.722
100x	0.825	0.680	0.680	0.777	0.793	0.848
200x	0.794	0.753	0.686	0.743	0.699	0.757
400x	0.731	0.761	0.659	0.773	0.701	0.731

Table 4. Test accuracy without cyclical learning rates decay

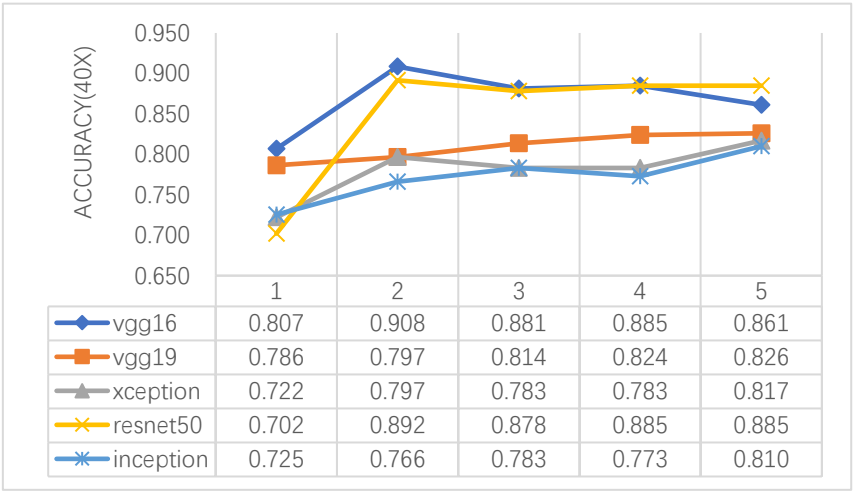

Figure 3. Results of 5 training sessions with different model cycles at 40x.

It can be seen from Table 5 that after CLR training, the accuracy of each model at 40x increases with the number of iterations in the test set, proving the effectiveness of continuous training.

Local optimal solutions for small data sets

We selected two of the eight types of breast cancer cells, benign selected for fibroadenoma and malignant selected for ductal carcinoma, for a total of benign malignancy. The test accuracies of two breast cancer cell types are shown in Table 6. These two breast cancers were selected because they represent a larger portion of the entire dataset and they are the representatives of the benign and malignant tumors correspondingly.

Magnification/Model	VGG16	VGG19	Xception	Resnet50	Inception	Inception_resnet
40x	0.946	0.886	0.783	0.952	0.994	0.970
100x	0.920	0.931	0.695	0.966	0.971	0.977
200x	0.906	0.953	0.947	0.959	0.982	0.959
400x	**0.986**	0.986	0.959	0.973	0.986	0.973

Table 6. Test accuracies of two breast cancer cell types (fibroadenoma and ductal_carcinoma).

From Table 6, we can see that the accuracy of the dichotomous classification task has improved significantly, and the average accuracy can reach more than 90% compared to the 8 classifications. In practical applications, we can jointly use a while classifier and 8 classifiers to get more accurate judgments.

Replacing softmax with logistic regression

To further improve the results, we used a logistic regression classifier, replacing the softmax layer. Since the softmax classifier is more suitable for datasets with more inter-class mutual exclusivity and less overlap, in the case of low similarity of features that cannot be model

extracted, we guess that using LR classifier will give better results.

	Vgg16	VGG19	Xception	Resnet50	Inception	Inception_resnet
40x	0.834 ↑	0.834 ↑	0.790 ↑	0.834 ↑	0.817 ↑	0.871 ↑
100x	0.828 ↓	0.822 ↑	0.738 ↓	0.828 ↓	0.770 ↑	0.882 ↑
200x	0.885 ↑	0.841 ↑	0.757↓	0.821 ↓	0.807↑	0.873 ↑
400x	0.845 ↑	0.826 ↑	0.727↓	0.848 ↓	0.754 ↑	0.923 ↑

Table 7. Table of the accuracy of the classification results of different models with LR. (↑ is higher than the original and ↓ is lower than the original)

Table 7 shows the accuracy results using LR. It can be seen that in most cases, LR improves the results to some extent.

IV. INTERFACE

To let the chatbot help the doctor predict the patient's condition, we generate the X.h5 to save the weights of the trained model. Then the trained models' architectures are saved in trained.py. Then we transfer parameters into the model automatically through the X.h5 files. Finally, the model can predict the processed image as follows.

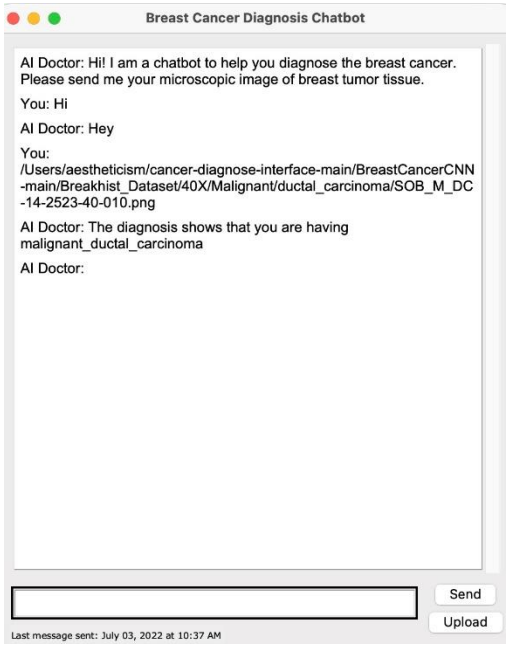

Figure 4. Dialog box interface for model loading and display of prediction results.

Specific implementation:

1. Imported Tkinter package to implement the chatbot.
2. The class "ChatInterface(Frame)" is individualized and built on a tkinter package by implementing a chat box.
3. Imported the nltk package to tokenize the 'breast cancer.txt'.
4. Added the upload button to help upload the image.
5. The chatbot can talk wisely thanks to the nltk package.

Finally, we can gather the codes of training inception_resnet_model and the interface, following the topological structure of data.

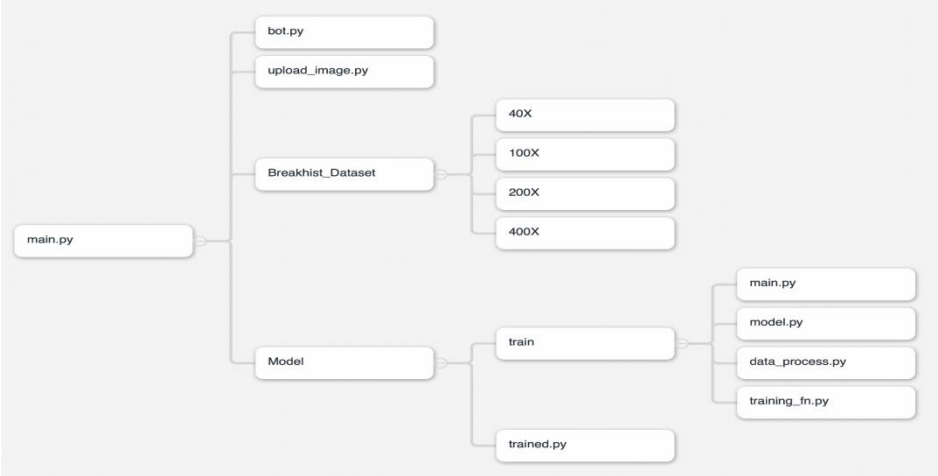

Figure 5. The code framework diagram of the interface part.

IV. DISCUSSION

Author(s)	Methodology	Dataset	Accuracy (in %)
Spanhol et al. (2015)	• Feature Extraction: using LBP, CLBP, LPQ, GLCM, PFTAS and ORB • Classification: 1-NN, QDA, SVM, and RF	BreaKHis	80%-85%
Spanhol et al. (2016)	• Pre-trained CNN Models: LeNet and AlexNet CNN	BreaKHis	80.8%-85.6%.
Bayramoglu et al. (2016)	• Proposed two CNN models: • Single task CNN: for predicting the malignancy. • Multi-task CNN: for predicting malignancy and image magnification levels	BreaKHis	80.6%-83.3%.
Spanhol et al. (2017)	• Extracted deep features (DeCAF): using pretrained CaffeNet.	BreaKHis	83.6%-84.8%.
Sudharshan et al. (2019)	• Proposed MIL based approach: APR, KNN, DD, SVM, non-parametric algorithm and MIL-CNN	BreaKHis	83.4%-92.1%.
Gour et al. (2020)	Developed a deep residual convolutional neural network (ResHist) for breast cancer diagnosis. • Proposed a data augmentation technique.	BreaKHis	84.34%-92.52%

Dabeer et al (2019)	7009 images from BreakHis database are used. Images captured are distributed into 4 magnification levels.	BreaKHis	99.86%
Bardou et al. (2018)	25% of the training data is used for cross validation. A 5 CNN layer topology with 3*3 filters and 2 fully connected layers is employed to classify the dataset. RELU layer is also applied.	BreaKHis	96.15% and 98.33% for the binary classification. 83.31% and 88.23% for multi-class classification.

Table 8. Paper review on the BreakHis dataset.

The accuracy we obtained so far is between 80%-90%, and we still have a lot of room for improvement in training and model building for this dataset, and the previous methods and accuracy rates using this dataset are listed in Table 8. In the future, we can build classification models using more powerful classifiers such as SVM, RF and XGBOOST. We can also make a deeper use of CLR in the tuning of learning rates. We should make greater use of models and datasets for more effective training.

Reference

Boyle, P., & Levin, B. (2008). *World cancer report 2008*. IARC Press.

He, K., Zhang, X., Ren, S., & Sun, J. (2016). Deep residual learning for image recognition. *2016 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*. https://doi.org/10.1109/cvpr.2016.90

Krizhevsky, A., Sutskever, I., & Hinton, G. E. (2017). ImageNet classification with deep convolutional Neural Networks. *Communications of the ACM*, *60*(6), 84–90. https://doi.org/10.1145/3065386

Satiate, A. (2019). *BreakHist-Dataset-Image-Classification*. GitHub. Retrieved July 1, 2022, from https://github.com/Anki0909/BreakHist-Dataset-Image-Classification

Simonyan, K., & Zisserman, A. (2014, December 19). *Very deep convolutional networks for large-scale image recognition*. arXiv.org. Retrieved July 2, 2022, from https://arxiv.org/abs/1409.1556v4

Spanhol, F. A., Oliveira, L. S., Petitjean, C., & Heutte, L. (2016). A dataset for breast cancer histopathological image classification. *IEEE Transactions on Biomedical Engineering*, *63*(7), 1455–1462. https://doi.org/10.1109/tbme.2015.2496264

Szegedy, C., Liu, W., Jia, Y., Sermanet, P., Reed, S., Anguelov, D., Erhan, D., Vanhoucke, V., & Rabinovich, A. (2015). Going deeper with convolutions. *2015 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*. https://doi.org/10.1109/cvpr.2015.7298594

x12hengyu. (2022). *Skin-Cancer-Classification-Chatbot*. GitHub. Retrieved July 1, 2022, from https://github.com/x12hengyu/Skin-Cancer-Classification-Chatbot

Appendix: Results of iterative training.

	vgg_16	vgg_19	xception	resnet50	inception	inception_resnet
40X	0.807	0.786	0.722	0.437	0.725	0.736
100X	0.845	0.777	0.744	0.880	0.735	0.764
200X	0.855	0.784	0.774	0.872	0.784	0.713
400X	0.826	0.803	0.754	0.883	0.735	0.750
40X	0.908	0.797	0.797	0.892	0.766	0.783
100X	0.874	0.819	0.790	0.922	0.783	0.822
200X	0.872	0.834	0.784	0.899	0.794	0.740
400X	0.833	0.792	0.723	0.856	0.693	0.739
40X	0.881	0.814	0.783	0.878	0.783	0.800
100X	0.893	0.845	0.767	0.890	0.757	0.848
200X	0.899	0.818	0.777	0.868	0.797	0.740
400X	0.833	0.807	0.678	0.837	0.712	0.750
40X	0.885	0.824	0.783	0.885	0.773	0.844
100X	0.893	0.858	0.767	0.900	0.780	0.864
200X	0.892	0.824	0.777	0.902	0.787	0.764
400X	0.822	0.780	0.678	0.833	0.670	0.780
40X	0.861	0.820	0.817	0.885	0.810	0.837
100X	0.896	0.838	0.786	0.896	0.774	0.819
200X	0.899	0.824	0.757	0.892	0.730	0.865
400X	0.803	0.784	0.720	0.845	0.769	0.841