Breast Cancer Image Classification using Convolutional Neural Networks

Julia Graf[1*†], Jana Hoffmann[1*†], Jessie Midgley[1*†] and Maike Nägele[1*†]

[1*]Algorithms in Bioinformatics, University of Tübingen, Sand 14, Tübingen, 72076, Germany.

*Corresponding author(s). E-mail(s): julia.graf@student.uni-tuebingen.de; jana2.hoffmann@student.uni-tuebingen.de; jessie.midgley@student.uni-tuebingen.de; maike.naegele@student.uni-tuebingen.de;
†These authors contributed equally to this work.

Abstract

Breast cancer, a frequently diagnosed and often life-threatening disease, poses a significant threat, particularly among females. The danger associated with breast cancer underscores the need for advanced techniques to improve the diagnosis of patients suffering from breast cancer. Mammography is a widely used technique for screening breast cancer. In this report, mammographic screens from the CBIS-DDSM Breast Cancer Image Dataset are used to perform binary breast cancer classification into malignant and benign cases. Classification is performed based on a machine learning approach using two different Convolutional Neural Networks (CNNs): VGG-16 and ResNet-50. ResNet-50 outperforms VGG-16 with an accuracy of 79.6% and and AUC of 71.9%.

1 Introduction

Breast cancer is a significant global health concern, impacting the lives of millions each year [1]. Early detection plays an important role in improving health outcomes and reducing mortality rates in cancer patients [2]. Currently, mammograms serve as a primary diagnostic tool in the identification of abnormalities in breast tissue [2]. However, the challenge of accurately interpreting these images and distinguishing between cancerous and non-cancerous abnormalities still remains [3]. The use of deep learning methods, particularly Convolutional Neural Networks (CNNs), have had a huge successes in medical image analysis [4]. Our project centers around the use of CNNs to classify mammograms into benign and malignant cases which, in theory, could aid healthcare professionals in making informed decisions about patient treatment plans. To train our CNNs we made use of the Curated Breast Imaging Subset of the Digital Database for Screening Mammography (CBIS-DDSM) dataset [5]. Since training a deep CNN requires a sufficiently large dataset in order to avoid overfitting, the issue of limited training samples is often addressed by transfer learning [6]. This technique uses a larger (often unrelated) database to train an initial model, which can then be fine-tuned on the dataset of interest [6]. Since large, publicly available medical imaging datasets are limited, we make use of the ImageNet dataset [7]. To develop our breast cancer classifier, we investigate the performance of two existing CNN architectures, namely VGG-16 [8] and ResNet-50 [9], and transfer these pre-learned weights to classify mammogram images. By systematically comparing the performance of these models, we seek to identify the most effective architecture for distinguishing between benign and malignant cases. In addition, we investigate the effectiveness of various image pre-processing and augmentation methods in improving model performance.

2 Materials and Methods

2.1 Dataset

The CBIS-DDSM dataset is an updated version of the Digital Database for Screening Mammography (DDSM) [10], containing digitized mammograms that have been selected and curated by a trained mammographer. We use a subset of 3.286 mammograms, containing 1.590 calcification cases and 1.696 mass cases, with a pathologic diagnosis for each image. We combine these cases and create a random train:test:validation split of 70:20:10.

2.2 Models

2.2.1 ResNet-50

ResNet-50 is a variant of Residual Network (ResNet) architecture, which makes use of skip connections designed to address the vanishing gradient problem [9]. The ResNet-50 architecture consists of 50 layers of weights and several residual learning blocks with varying number of convolutional layers within each block. Each convolutional layer uses batch normalization as a regularization method. In total, ResNet-50 is made up

of a total of 48 convolutional layers, along with 2 pooling layers, producing a grand total of 23 million trainable parameters. The network takes images as input with a dimension of 224 × 224 pixels, and uses softmax in the final layer for the classification of input images.

2.2.2 VGG-16

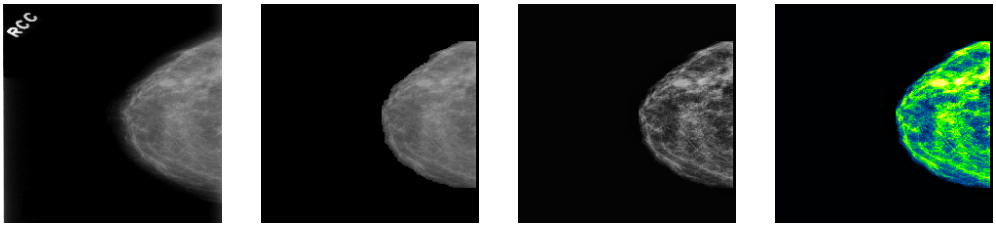
VGG-16 is an architecture proposed by Simonyan and Zisserman in 2014, containing 16 layers [11]. It consists of different types of layers and also takes RGB images of size 224 × 224 pixels. The early layers in the network comprise 13 convolution layers containing 5 max-pooling layers. This is followed by a dense layer flattening the output of the previous layers and applying a non-linear sigmoid to perform the actual classification. In total, VGG-16 has around 15 million trainable parameters [12].

2.3 Optimizing the model

2.3.1 Keeping Aspect Ratio

Since the mammograms do not have the dimensions required for the models, we resize them. The built-in `resize` function from `scikit-image` does not keep the aspect ratio of pictures [13]. Since this can distort important features of the image, we implemented a function that keeps the aspect ratio unchanged while resizing the images. Any missing parts are filled in with black to ensure the image remains at the desired dimensions. We compare the performance of the models on images of these two resizing methods.

2.3.2 Preprocessing Images



(a) Resizing of images (b) Artefacts Removal (c) Image Enhancement (d) Green Fire Blue Filter

Fig. 1: Image preprocessing pipeline

To process the images, we implement a workflow that was proposed by Montaha et al. [14]. Sometimes, undesired objects or thin white borders appear in the images as shown in Figure 1a. As these can impair the performance of the model, we remove them. Firstly, the white borders are removed by coloring a border of 5 pixels black. This way, the borders are removed without losing necessary information. In the next

step, to eliminate noise from the pictures, we binarize them and apply morphological opening [15]. To get rid of any remaining artefacts, we apply largest contour detection [16]. Finally, the resulting binary mask is merged with the original image, as shown in Figure 1b. In mammograms, fatty tissues appear dark while both dense breast tissues and tumors have a light appearance [14]. This means that tumors are difficult to detect in dense tissues. To enhance the contrast, we apply gamma correction and `CLAHE`, which was developed for medical imaging to improve the visibility of complex structures [14] (Figure 1c). The final step in image preprocessing is applying the `ImageJ` Green Fire Blue (GFB) filter on the image to separate the tumor, dense tissues and surrounding cells in three different colors (see Figure 1d).

2.3.3 Fine Tuning

We experiment with re-training the last layer of the pre-trained model with the new images. This means that the original weights from the first to the penultimate layer are preserved or "frozen", and the last layer is replaced with new weights in order to learn task-specific features. In addition, a final fully-connected layer with 128 units was added to the ResNet-50 model in order to increase the capacity.

2.3.4 Optimizer and Learning Rate

We compare the performance of two different optimizers, namely Adam and Nadam. Adam computes individual adaptive learning rates for different parameters from estimates of first and second moments of the gradients [17]. Nadam is an extension of the Adam optimizer that incorporates Nesterov momentum and can lead to improved convergence [18]. The learning rate is a hyperparameter of the optimizer, determining the weight with which parameters are updated at each iteration during optimization [19]. We compare the two optimization algorithms at three different learning rates. With a larger learning rate, the model tends to converge faster, however with an increased risk of overshooting the optimal value of the loss function [19]. If the learning rate is too small, the training process might be time-consuming.

2.3.5 Flatten vs. Global Average Pooling

In order to pass the output of the last convolutional layer to the fully-connected layer and then further to the output layer, the vector needs to be converted into a one-dimensional vector. This can be done by applying a Flatten layer where the resulting one-dimensional vector contains the same values as before the flattening. Another layer that we can add to reduce the dimensions is a Global Average Pooling layer. This layer creates one feature map for each class of the output and the average of each is passed to the last layer [20].

2.3.6 Image Data Generator

After splitting the dataset into train, validation, and test sets, we are left with 2.300 training images. We compare the performance of our model on these training images to the images of the train set randomly modified by tensorflow's `ImageDataGenerator`

before each epoch [21]. We set the parameters, so that the input images are randomly flipped by a range between 0° and 40°, shifted on their horizontal axis up to 20% of their width, shifted on their vertical axis up to 20% of their height, zoomed by a factor within a range of 0.8 and 1.2, flipped from left to right and the shear transformation is applied to them with an angle of 0.2°. The fill mode fills pixels outside as the nearest pixel. The number of images per epoch stays unchanged.

2.3.7 Data Augmentation

We also test the performance using an augmentation method that increases the amount of images the model sees in each epoch. We therefore implemented the augmentation techniques proposed by Montaha et al. [14]. Through the augmentation we generate seven new images for every original image by flipping and rotating. After the augmentation the train set includes 18.400 images.

3 Results

3.1 Evaluation strategy

To find the best performing models, we optimized for validation accuracy and AUC. Different combinations of various custom classification layers, optimizers, learning rates, and differently preprocessed image sets were explored.
To evaluate the best models, we train on five different train/validation splits, created by applying the `train_test_split` function from `scikit-learn` [22]. In the end we plot the ROC-curve for each of the runs and compute the mean test accuracy and the mean test AUC.

3.2 VGG-16

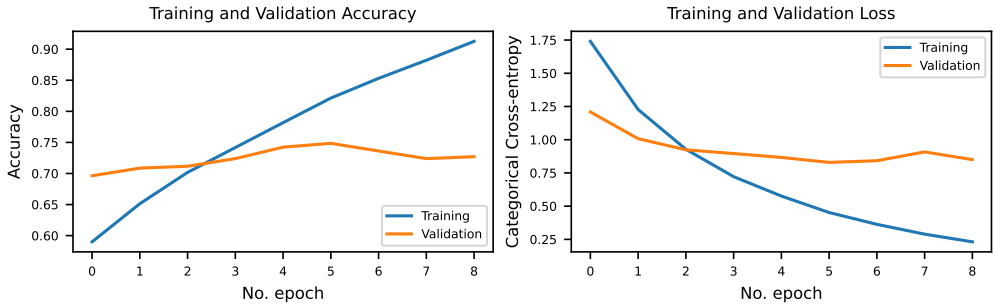


Fig. 2: Learning curves of the best-performing VGG-16 model.

Table 1 displays the performance measures of the VGG-16 optimization trials. The best model is highlighted in the table. It uses a learning rate of 0.00001, the Adam optimizer, and a Flatten layer, resulting in a validation accuracy of 72.3% and an AUC of 80.1%. Figure 2 shows that training and validation accuracy increase, whereas

Table 1: Results of the VGG-16 model optimization

Prepro	Last Layer Trainable	Adam/ Nadam	Flatten/ GAP[1]	Learning rate	Data Augm.[2]	H/W Ratio[3]	Acc[4]	AUC[5]
built-in	no	Adam	Flatten	0.0001	no	no	0.6933	0.7353
built-in	yes	Adam	Flatten	0.0001	no	no	0.6656	0.7487
self-written	no	Adam	Flatten	0.0001	no	no	0.5920	0.6708
built-in	no	Nadam	Flatten	0.0001	no	no	0.6656	0.7337
built-in	**no**	**Adam**	**Flatten**	**0.00001**	**no**	**no**	**0.7270**	**0.8016**
built-in	no	Adam	Flatten	0.001	no	no	0.6748	0.7034
built-in	no	Adam	GAP	0.00001	no	no	0.5982	0.6355
built-in	no	Adam	Flatten	0.00001	yes	no	0.7086	0.7487
built-in	no	Adam	Flatten	0.00001	no	yes	0.7117	0.7286

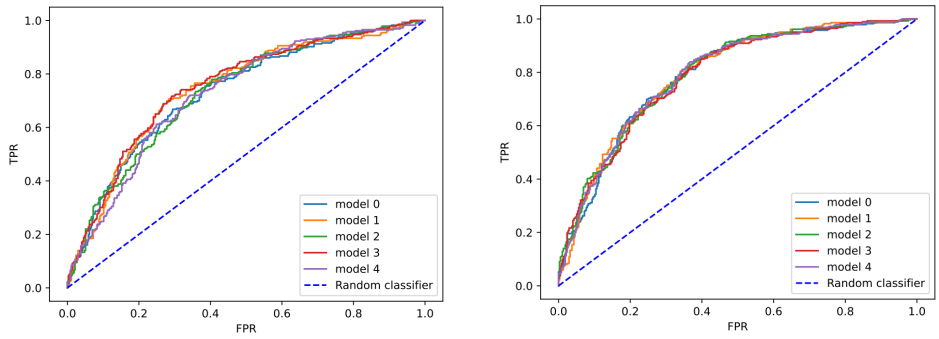
[1]Global Average Pooling

[2]using built-in Image Data Generator

[3]Keeping height/width ratio

[4]Accuracy on validation set

[5]AUC on validation set



(a) VGG-16 (b) ResNet-50

Fig. 3: ROC curves of the best models trained on different training/validation splits.

the loss decreases. The validation of the best model shows that each split results in a similar AUC as seen in Figure 3a. The model with best performance achieves a mean accuracy of 68.6% and a mean AUC of 73.7% over all splits.

3.3 ResNet-50

Table 2 shows the performance measures of the ResNet-50 optimization trials. The combination that achieved the best result is highlighted in the table. It uses a learning rate of 0.0001, the Nadam optimizer, and a Global Average Pooling layer, resulting in a validation accuracy of 75.2% and an AUC of 82.7%. Figure 4 shows the correspond-

Table 2: Results of the ResNet-50 model optimization

Prepro	Last Layer Trainable	Adam/ Nadam	Flatten/ GAP[1]	Learning rate	Data Augm.	H/W Ratio[4]	Acc[5]	AUC[6]
built-in	yes	Nadam	Flatten	0.0001	no	no	0.7209	0.8036
self-written	yes	Nadam	Flatten	0.0001	no	no	0.6564	0.7217
built-in	no	Nadam	Flatten	0.0001	no	no	0.7270	0.8234
built-in	**no**	**Nadam**	**GAP**	**0.0001**	**no**	**no**	**0.7515**	**0.8269**
built-in	no	Nadam	GAP	0.001	no	no	0.7086	0.7959
built-in	no	Nadam	GAP	0.00001	no	no	0.6871	0.7617
built-in	no	Nadam	GAP	0.0001	yes [2]	no	0.5613	0.5914
built-in	no	Nadam	GAP	0.0001	yes [3]	no	0.6902	0.7486
built-in	no	Nadam	GAP	0.0001	no	yes	0.6902	0.7705
built-in	no	Adam	GAP	0.0001	no	no	0.7515	0.8128

[1]Global Average Pooling

[2]Using self-written Data Augmentation

[3]Using built-in Image Data Generator

[4]Keeping height/width ratio

[5]Accuracy on validation set

[6]AUC on validation set

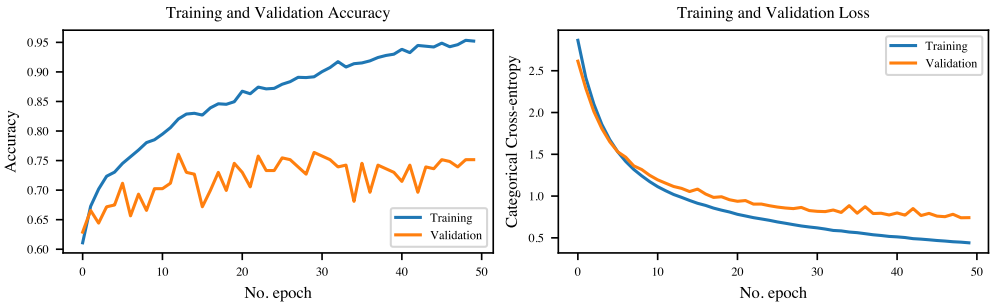


Fig. 4: Learning curves of the best-performing ResNet-50 model.

ing learning curves. Figure 3b shows the ROC curves for all five models. The mean accuracy of all five models was 79.6% and the mean AUC was 71.9%.

4 Discussion

Despite using a workflow similar to that proposed by Montaha et al., we were not able to achieve their accuracy of 98.02% for the VGG-16 model. We assume their high accuracy resulted from them augmenting the data before splitting it. This way, it is possible that the model is tested on images of which it may have already seen augmented versions during training. The authors also explored the ResNet-50 architecture but achieved worse results. For us, the ResNet-50 model outperformed the VGG-16 model. We hypothesize that this is due to its deeper architecture and the incorporation of residual connections, which address the vanishing gradient problem. Additionally, since training is a stochastic process and we only performed one training round for each optimization variable, it is possible that the actual optimum is not represented in the metrics we chose to define the best model. Retrospectively, we think it would boost performance to combine data augmentation with more trainable parameters.

5 Conclusion

Two existing CNN architectures, namely VGG-16 and ResNet-50 are used to classify mammograms into benign and malignant. Additionally, data augmentation, data preprocessing, and hyperparameter tuning are presented as important steps to improve predictions and achieve generalized models. Performance analysis of the models indicates the possibilities and benefits of artificial neural networks in medical image classification. For both models, an AUC of over 70% is achieved. Although, the values indicate a moderate performance it is important to consider the limitations for further improvements.

5.1 Limitations

The CBIS-DDSM dataset, used for classification, consists of 3.286 images. For a model to perform good classification the size of the available training data plays an important role. As the data is taken from a publicly available source, there is a limited amount of data available. Finally, hyperparameter tuning was restricted due to time and memory issues. As the servers were not accessible during the time of model training, all the hyperparameter tuning was run locally leading to long runtimes and memory-related issues.

References

[1] Siegel, R.L., Miller, K.D., Wagle, N.S., Jemal, A.: Cancer statistics, 2023. CA: A Cancer Journal for Clinicians **73**(1), 17–48 https://doi.org/10.3322/caac.21763

[2] Mathew, J., Sibbering, M.: In: Wyld, L., Markopoulos, C., Leidenius, M., Senkus-Konefka, E. (eds.) Breast Cancer Screening, pp. 147–156. Springer, Cham (2018)

[3] Elter, M., Horsch, A.: Cadx of mammographic masses and clustered micro-calcifications: A review. Medical Physics **36**(6Part1), 2052–2068 (2009) https://doi.org/10.1118/1.3121511

[4] Cai, L., Gao, J., Zhao, D.: A review of the application of deep learning in medical image classification and segmentation. Ann. Transl. Med. **8**(11), 713 (2020)

[5] Lee, R.S., Gimenez, F., Hoogi, A., Miyake, K.K., Gorovoy, M., Rubin, D.L.: A curated mammography data set for use in computer-aided detection and diagnosis research. Sci. Data **4**(1), 170177 (2017)

[6] Kim, H.E., Cosa-Linan, A., Santhanam, N., Jannesari, M., Maros, M.E., Ganslandt, T.: Transfer learning for medical image classification: a literature review. BMC Med. Imaging **22**(1), 69 (2022)

[7] Deng, J., Dong, W., Socher, R., Li, L.-J., Li, K., Fei-Fei, L.: ImageNet: A large-scale hierarchical image database. IEEE (2009)

[8] Simonyan, K., Zisserman, A.: Very Deep Convolutional Networks for Large-Scale Image Recognition (2015)

[9] He, K., Zhang, X., Ren, S., Sun, J.: Deep Residual Learning for Image Recognition (2015)

[10] Heath, M.D., Bowyer, K., Kopans, D.B., Moore, R.H.: The digital database for screening mammography. (2007). https://api.semanticscholar.org/CorpusID:68362967

[11] Alzubaidi, L., Zhang, J., Humaidi, A.J., Al-Dujaili, A.Q., Duan, Y., Al-Shamma, O., Santamaría, J., Fadhel, M.A., Al-Amidie, M., Farhan, L.: Review of deep learning: concepts, cnn architectures, challenges, applications, future directions. Journal of Big Data **8**(1) (2021) https://doi.org/10.1186/s40537-021-00444-8

[12] Radhika, K., Devika, K., Aswathi, T., Padma, S., Vishvanathan, S., Kp, S.: Performance Analysis of NASNet on Unconstrained Ear Recognition, pp. 57–82 (2020). https://doi.org/10.1007/978-3-030-33820-6_3

[13] skimage.transform. https://scikit-image.org/docs/stable/api/skimage.transform.html#skimage.transform.resize. Accessed: 2024-01-23

[14] Montaha, S., Azam, S., Rafid, A.K.M.R.H., Ghosh, P., Hasan, M.Z., Jonkman, M., De Boer, F.: BreastNet18: A High Accuracy Fine-Tuned VGG16 Model Evaluated Using Ablation Study for Diagnosing Breast Cancer from Enhanced Mammography Images. Biology **10**(12) (2021) https://doi.org/10.3390/biology10121347.PMID:34943262

[15] cv2.morphologyEx. https://docs.opencv.org/4.x/d9/d61/tutorial_py_morphological_ops.html. Accessed: 2024-01-30

[16] cv2.findContours. https://docs.opencv.org/3.4/d3/dc0/group__imgproc__shape.html#ga17ed9f5d79ae97bd4c7cf18403e1689a. Accessed: 2024-01-30

[17] Kingma, D.P., Ba, J.: Adam: A Method for Stochastic Optimization (2017)

[18] Dozat, T.: Incorporating Nesterov Momentum into Adam. In: Proceedings of the 4th International Conference on Learning Representations, pp. 1–4

[19] Brownlee, J.: Understand the impact of learning rate on neural network performance (2020). https://machinelearningmastery.com/understand-the-dynamics-of-learning-rate-on-deep-learning-neural-networks/

[20] Admin: Explain Pooling layers: Max Pooling, Average Pooling, Global Average Pooling, and Global Max pooling. https://androidkt.com/explain-pooling-layers-max-pooling-average-pooling-global-average-pooling_-and-global-max-pooling/ (2023)

[21] tf.keras.preprocessing.image.ImageDataGenerator. https://www.tensorflow.org/api_docs/python/tf/keras/preprocessing/image/ImageDataGenerator. Accessed: 2024-01-23

[22] sklearn.model_selection.train_test_split. https://scikit-learn.org/stable/modules/generated/sklearn.model_selection.train_test_split.html. Accessed: 2024-01-25