RESEARCH

Introduction to Focus Areas in Bioinformatics Project Week 3

Sina Glöckner[*†], Christina Kirschbaum[†], Swetha Rose Maliyakal Sebastian[†] and Gokul Thothathri[†]

[*]Correspondence:
sina.gloeckner@fu-berlin.de
Institute for Informatics, Freie Universität Berlin, Takustr. 9, Berlin, DE
Full list of author information is available at the end of the article
[†]Equal contributor

Abstract

Goal of the project: The goal was to apply three different deep learning algorithms to the BreaKHis data set, to evaluate and compare them.

Main results of the project: Overall, the CNN performed best, because the RNN and the MLP overfitted.

Personal key learnings:
Sina: I gained new information about breast cancer, and learned strategies to analyze and understand foreign code.
Christina: Differences in preprocessing of the data set when images are used
Swetha: Multilayer Perceptron, understood image processing using deep learning techniques.
Gokul: I learnt RNN using LSTM deeplearning strategies, understood how to create a model for image classification.

Estimation of the time:
Sina: 14 hours
Christina: 14 hours
Swetha: 13 hours
Gokul: 13 hours

Project evaluation: 4

Number of words: 1309

1 Scientific Background

Female breast cancer was the most commonly diagnosed cancer in 2020 with an estimated 2.3 million new cases. It is responsible for 6.9% of cancer deaths [1]. The awareness of the disease has risen in the last 30 years, and simultaneously the research has become more extensive [2]. To reliably diagnose breast cancer a combination of multiple imaging methods, such as mammography, ultrasonography, or biopsies, are used [3]. One of the three different types of biopsies, the excisional biopsy, is the focus of this report.

Since excisional biopsies, also called the SOB method, require anesthesia and are done in a hospital, this method is only used if less intrusive approaches had inconclusive results [3]. During the procedure, samples of cells or tissues are collected and fixed on a glass microscope slide to be stained and examined [4]. The manual evaluation of the sample is time-consuming. Therefore, the demand for computer-assisted diagnostic algorithms is high [5]. With the help of deep learning algorithms, it is possible to classify image data based on an existing data set. The BreaKHis database includes such a data set.

2 Goal

In this project, we apply three different deep learning algorithms to the BreaKHis data set. Each algorithm is evaluated and compared based on its accuracy and loss in five epochs.

3 Data and Preprocessing

BreaKHis is composed of 7909 clinically representative microscopic images of breast cancer tissue from 82 patients. The database contains 2480 samples of benign tumors and 5429 samples of malignant tumors. The tumor samples were sorted into eight histologically distinct types of cancer. During our analysis, we ignored this information. Additionally, four different magnifying factors are part of the data set. To filter the data, we used files with a magnifying factor of 100X. [4]

For every image in the data set, we went through several preprocessing steps using the OpenCV library [6]. After reading the images, we converted their color space to the common RGB space. To further standardize the images, each picture was resized to a uniform size. The images were then smoothed with a median filter of 5 and lastly equalized in the HSV color space.

We split this data set into two parts: a validation set and a training set. The training data set consists of 1666 images, while the validation batch consists of 415 pictures. For this process and the following algorithms, we employed the python library TensorFlow [7].

4 Methods

We applied three different deep learning algorithms to the data set and evaluated and compared them using two different evaluation metrics.

4.1 Convolutional Neural Network

The first algorithm we used was a convolutional neural network (CNN). Figure 1 (left) shows all layers used in our computation. We first applied six convolutions, then flattened the model. Lastly, we applied alternating dense, and dropout layers.

This model is the most common for image analysis.

4.2 Recurrent Neural Network using LSTM

Our second network was a recurrent neural network (RNN). In these networks, a temporal dimension is added. To put it another way, the network's prediction from the first run is used as an input in the second run. Multiple RNNs, each trained for a specific job, are controlled by logic gates in long short-term memory (LSTM) [8].

In our case, we applied TensorFlows LSTM twice and further classified the data with dense and dropout (Figure 1, right).

4.3 MLP

For our last model, we used the multi-layer perception technique (MLP). This is a class of feed-forward artificial neural network [9]. MLPs are constructed using mathematical neurons and their synapses, which are called weights in this case.

In this case, one layer was to flatten the data, then two dense functions were applied (Figure 1, middle). This network is not deep, since it only consists of three layers.

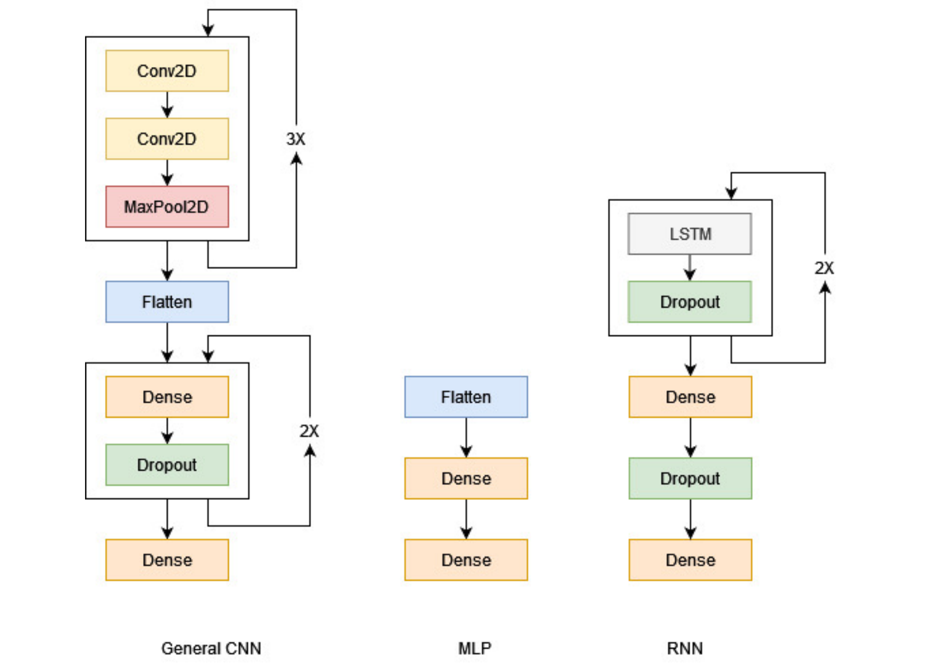


Figure 1: **Layers of custom models** The different layers used in our CNN model (left) and our MLP model (right) are shown here.

4.4 Evaluation Metrics

We evaluated and compared these three algorithms with the help of Loss and Accuracy. To that end, these two metrics during five epochs of each model on training and validation data were documented.

The Accuracy specifies the correctly identified positive and negative samples.

$$Accuracy = \frac{\Sigma \text{ True Positives} + \Sigma \text{ True Negatives}}{\Sigma \text{ Positives} + \Sigma \text{ Negatives}}$$

Coupled with the Loss, we can accurately assess the model. The Loss analyses the difference between the prediction and the reality in logarithmic space.

$$Loss = -\frac{1}{N}\sum_{n=1}^{N}[(\text{True Result})log(\text{Predicted Result}) + (1 - \text{True Result})log(1 - \text{Predicted Result})]$$

Further, we analyze the models in different epochs. This means the whole data set is only forward and backward through the neural network once. We split one epoch into numerous smaller batches to minimize computational costs. The number of epochs equals the number of iterations if the batch size is the entire training data set.

5 Results

The RNN showed an accuracy 68.61% in the first epoch of the training set, 69.03% in the other epochs and a constant accuracy of 69.16% in the validation set. The loss sank from 0.63 to 0.57 in the training set. In the validation set, the loss had a range from 0.54 and 0.61. The results are visualized in Figure 2.

For the MLP, the accuracy ranged between 67.53% and 84.03% and the loss between 5.70 and 1.18 in the training set. The values for the validation set alternate. In epoch two and five, the accuracy fell to 41.13% and the loss jumped to 7.75 and 6.87. A summary of the results is shown in Figure 3.

In the first epoch, the CNN had an accuracy of 66.15% and a loss of 0.62 for the training set. By the fifth epoch, the accuracy increased to 72.93% and the loss decreased to 0.51. The validation set reached values between 69.16% and 70.12% for accuracy as well as 0.58 and 0.51 for loss. In Figure 4 the accuracy and loss over all five epochs can be seen.

Overall, the CNN performed best. The accuracy increased and the loss decreased nearly uniformly. This shows the best learning progress.

6 Discussion

Originally, we planned to use the InceptionV3 network and compare it to the CNN. However, the code is required to run on Google Colab and because of its limitations this was not possible.

Therefore, a CNN, a RNN, and a MLP were implemented. Due to over-fitting of RNN and MLP, these deep learning networks did not perform well. All samples of the validation batch were classified as malignant tumors. The reason for this is the imbalanced data set, that contained twice as much malignant than benign samples. In further approaches, that could be prevented through sampling methods which are applied to the data set. Another option is to better normalize the data before applying the neural networks.

The project incorporates various tasks which are typical for a data scientist. The data processing and transformation included new approaches because of the work with images instead of text data. This also required another way of thinking to understand the data. Additionally, programming and an understanding for different deep learning networks were required.

Competing interests
The authors declare that they have no competing interests.

Author's contributions
Swetha Rose Maliyakal Sebastian implemented the MLP model. Gokul Thothathri implemented the RNN model. Sina Glöckner informed about the scientific background and wrote the report. Christina Kirschbaum conducted the preprocessing, and implemented the CNN model.

References

1. Sung, H., Ferlay, J., Siegel, R.L., Laversanne, M., Soerjomataram, I., Jemal, A., Bray, F.: Global Cancer Statistics 2020: GLOBOCAN Estimates of Incidence and Mortality Worldwide for 36 Cancers in 185 Countries. CA: A Cancer Journal for Clinicians **71**(3), 209–249 (2021). doi:10.3322/caac.21660. _eprint: https://onlinelibrary.wiley.com/doi/pdf/10.3322/caac.21660. Accessed 2021-11-11
2. Braun, S.: The History of Breast Cancer Advocacy. The Breast Journal **9**(s2), 101–103 (2003). doi:10.1046/j.1524-4741.9.s2.13.x. _eprint: https://onlinelibrary.wiley.com/doi/pdf/10.1046/j.1524-4741.9.s2.13.x. Accessed 2021-11-11
3. Apantaku, L.M.: Breast cancer diagnosis and screening. Am Fam Physician **62**(3), 596–602605606 (2000)
4. Spanhol, F.A., Oliveira, L.S., Petitjean, C., Heutte, L.: A Dataset for Breast Cancer Histopathological Image Classification. IEEE Transactions on Biomedical Engineering **63**(7), 1455–1462 (2016). doi:10.1109/TBME.2015.2496264. Conference Name: IEEE Transactions on Biomedical Engineering
5. Gurcan, M.N., Boucheron, L.E., Can, A., Madabhushi, A., Rajpoot, N.M., Yener, B.: Histopathological Image Analysis: A Review. IEEE Reviews in Biomedical Engineering **2**, 147–171 (2009). doi:10.1109/RBME.2009.2034865. Conference Name: IEEE Reviews in Biomedical Engineering

6. Bradski, G.: The OpenCV Library. Dr. Dobb's Journal of Software Tools (2000)
7. Abadi, M., Agarwal, A., Barham, P., Brevdo, E., Chen, Z., Citro, C., Corrado, G.S., Davis, A., Dean, J., Devin, M., Ghemawat, S., Goodfellow, I., Harp, A., Irving, G., Isard, M., Jia, Y., Jozefowicz, R., Kaiser, L., Kudlur, M., Levenberg, J., Mané, D., Monga, R., Moore, S., Murray, D., Olah, C., Schuster, M., Shlens, J., Steiner, B., Sutskever, I., Talwar, K., Tucker, P., Vanhoucke, V., Vasudevan, V., Viégas, F., Vinyals, O., Warden, P., Wattenberg, M., Wicke, M., Yu, Y., Zheng, X.: TensorFlow: Large-Scale Machine Learning on Heterogeneous Systems. Software available from tensorflow.org (2015). https://www.tensorflow.org/
8. Lipton, Z.: A critical review of recurrent neural networks for sequence learning (2015)
9. Friedman, J.H.: The Elements of Statistical Learning: Data Mining, Inference, and Prediction. springer open, ??? (2017)

Figures

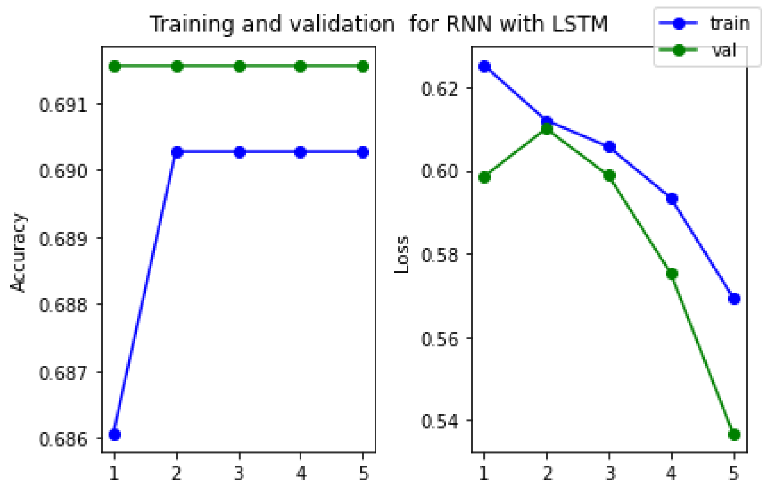


Figure 2: **Results for RNN** Here the accuracy and loss for our RNN model in five epochs can be seen.

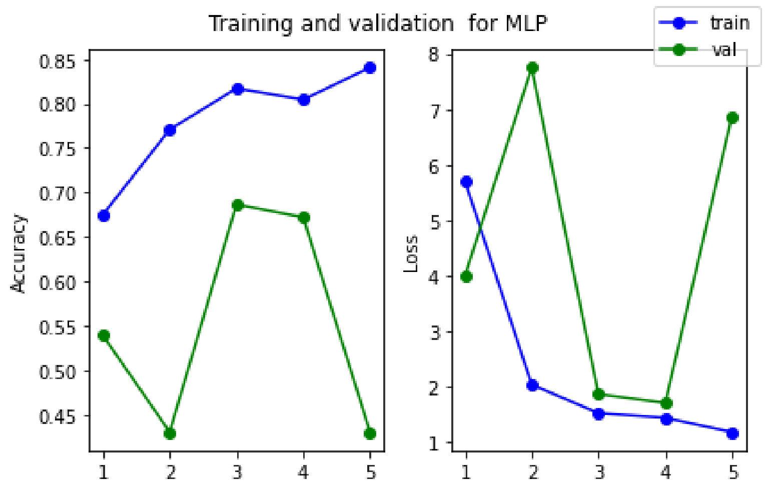


Figure 3: **Results for MLP** Here the accuracy and loss for our MLP model in five epochs can be seen.

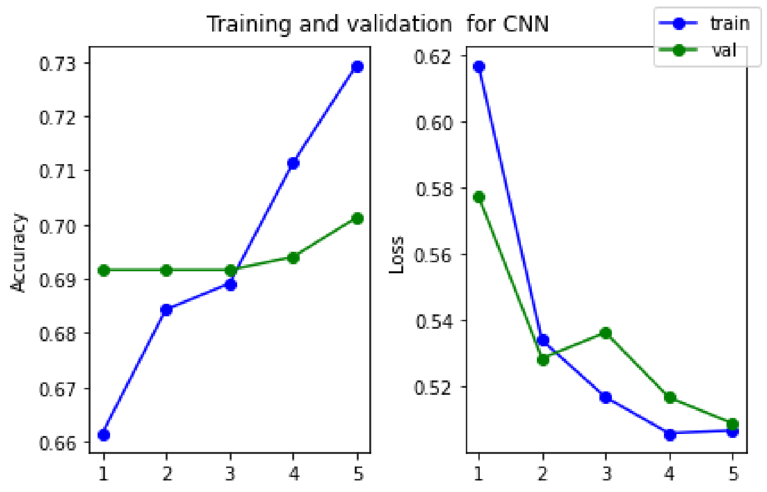
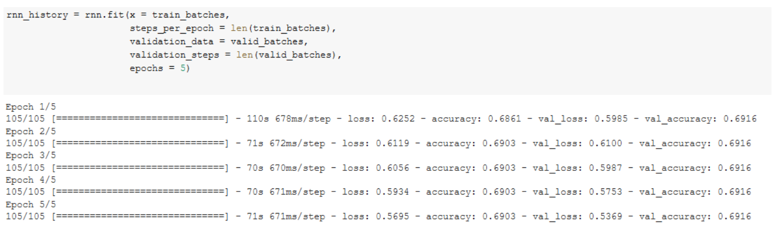


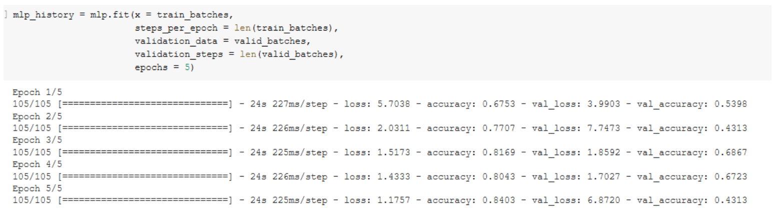
Figure 4: **Results for CNN** Here the accuracy and loss for our CNN model in five epochs can be seen.



```
rnn_history = rnn.fit(x = train_batches,
                      steps_per_epoch = len(train_batches),
                      validation_data = valid_batches,
                      validation_steps = len(valid_batches),
                      epochs = 5)

Epoch 1/5
105/105 [==============================] - 110s 678ms/step - loss: 0.6252 - accuracy: 0.6861 - val_loss: 0.5985 - val_accuracy: 0.6916
Epoch 2/5
105/105 [==============================] - 71s 672ms/step - loss: 0.6119 - accuracy: 0.6903 - val_loss: 0.6100 - val_accuracy: 0.6916
Epoch 3/5
105/105 [==============================] - 70s 670ms/step - loss: 0.6056 - accuracy: 0.6903 - val_loss: 0.5987 - val_accuracy: 0.6916
Epoch 4/5
105/105 [==============================] - 70s 671ms/step - loss: 0.5934 - accuracy: 0.6903 - val_loss: 0.5753 - val_accuracy: 0.6916
Epoch 5/5
105/105 [==============================] - 71s 671ms/step - loss: 0.5695 - accuracy: 0.6903 - val_loss: 0.5369 - val_accuracy: 0.6916
```

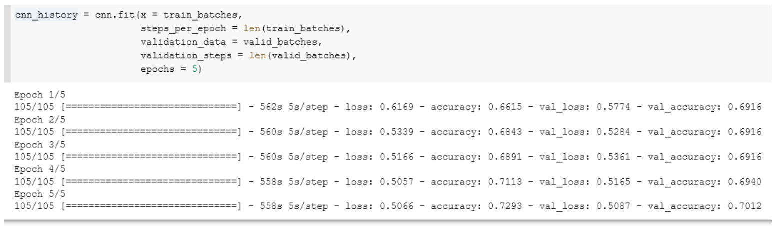
Figure 5: **Screenshot** This Screenshot shows the final classifier results for RNN.



```
] mlp_history = mlp.fit(x = train_batches,
                      steps_per_epoch = len(train_batches),
                      validation_data = valid_batches,
                      validation_steps = len(valid_batches),
                      epochs = 5)

Epoch 1/5
105/105 [==============================] - 24s 227ms/step - loss: 5.7038 - accuracy: 0.6753 - val_loss: 3.9903 - val_accuracy: 0.5398
Epoch 2/5
105/105 [==============================] - 24s 226ms/step - loss: 2.0311 - accuracy: 0.7707 - val_loss: 7.7473 - val_accuracy: 0.4313
Epoch 3/5
105/105 [==============================] - 24s 225ms/step - loss: 1.5173 - accuracy: 0.8169 - val_loss: 1.8592 - val_accuracy: 0.6867
Epoch 4/5
105/105 [==============================] - 24s 226ms/step - loss: 1.4333 - accuracy: 0.8043 - val_loss: 1.7027 - val_accuracy: 0.6723
Epoch 5/5
105/105 [==============================] - 24s 225ms/step - loss: 1.1757 - accuracy: 0.8403 - val_loss: 6.8720 - val_accuracy: 0.4313
```

Figure 6: **Screenshot** This Screenshot shows the final classifier results for MLP.



```
cnn_history = cnn.fit(x = train_batches,
                      steps_per_epoch = len(train_batches),
                      validation_data = valid_batches,
                      validation_steps = len(valid_batches),
                      epochs = 5)

Epoch 1/5
105/105 [==============================] - 562s 5s/step - loss: 0.6169 - accuracy: 0.6615 - val_loss: 0.5774 - val_accuracy: 0.6916
Epoch 2/5
105/105 [==============================] - 560s 5s/step - loss: 0.5339 - accuracy: 0.6843 - val_loss: 0.5284 - val_accuracy: 0.6916
Epoch 3/5
105/105 [==============================] - 560s 5s/step - loss: 0.5166 - accuracy: 0.6891 - val_loss: 0.5361 - val_accuracy: 0.6916
Epoch 4/5
105/105 [==============================] - 558s 5s/step - loss: 0.5057 - accuracy: 0.7113 - val_loss: 0.5165 - val_accuracy: 0.6940
Epoch 5/5
105/105 [==============================] - 558s 5s/step - loss: 0.5066 - accuracy: 0.7293 - val_loss: 0.5087 - val_accuracy: 0.7012
```

Figure 7: **Screenshot** This Screenshot shows the final classifier results for CNN.