RESEARCH

Introduction to Focus Areas - Project 1 - Group 8

Dominik Bannwitz*, Florian Herzler, Maximilian Otto and Mariana Steffens

*Correspondence:
dominik.bannwitz@fu-berlin.de
Department of Mathematics and Computer Science, Free University of Berlin, Takustraße 9, 14195 Berlin, Germany
Full list of author information is available at the end of the article
†Equal contributor

Abstract

Goal of the project: Perform an exploratory analysis and develop three classifiers for predicting the diagnose of heart disease.

Methods used in the project: As classification models, Logistic Regression, K-nearest neighbor and Decision Trees were implemented and evaluated.

Main results of the project: The Logistic Regression classifier was accurate at predicting the target groups with an accuracy of about 87%.

Personal key learnings: We implemented the classifiers Logistic Regression, K-nearest neighbor and Decision Trees for the first time.

Estimated working hours: 54

Possible improvements: Techniques to avoid overtraining of Decision Trees can be explored.

Keywords: Classifiers; Logistic Regression; K-nearest Neighbor; Decision Tree

1 Scientific Background

Computers are being used to support decision-making in more and more areas. In some, they have already taken over the decisions completely - in others, it is hard to imagine what the decision would be like without their support. Taking the example of autonomous driving, we can observe what decisions the computer has to make. The car's camera recognizes a traffic signal and must now decide whether it should stop at the intersection or whether it can drive over it. To be able to decide this, the car's computer must have previously learned what license plates exist and now classify the seen license plate as one of them. To understand how this works, we will introduce the concept of classification.

In data science, classification is the process of predicting which of a set of classes an observation belongs to. Many different algorithms for classification have been developed over the years. There are more "traditional" classification approaches such as k-nearest neighbor(KNN), logistic/linear regression, or decision trees, as well as more "modern" algorithms such as neural networks and deep learning. What all these methods have in common is that they must first be trained on a training data set before they can be evaluated on another test data set. Therefore, a considerable amount of data must be accumulated and categorized by some other program or expert. The data set must then be divided into a training and test data set, and a classifier must be chosen, considering its characteristics, benefits and limitations. The process of training the classifier often requires repetitions to adjust weights or other parameters until a certain accuracy is achieved or a set number of repetitions is reached. Once the classifier is trained, testing is required. To do this, we have to run the learned classifier on the test dataset and then choose sensible evaluation parameters. For example, recall and precision or the F1-index, as well as the ROC-Curve (Receiver Operating Characteristic) and AUC (Area under Curve) are often used to evaluate the ability of the trained classifier.

2 Goal

The goal was to perform an exploratory analysis and develop three different classifiers for diagnosing heart disease based on the "Cleveland Heart Disease" data set, and then analyze and compare the performance of each classifier.

3 Data and Preprocessing

The data was collected from 303 patients referred for coronary angiography disease (CAD) at the Cleveland Clinic, in Cleveland, Ohio, between May 1981 and September 1984 [1]. The original data set is available at the "UCI Machine Learning Repository"[1] and contains 303 instances and 76 attributes of which 14 were used. The "goal" field refers to the presence of heart disease and is an integer value from 0 (no presence) to 4.
As a first step of the data preprocessing, the table was read as a data frame and all the values were converted to float, except the column "goal", which is categorical data and was converted to string for better visualization. The next step was to find out the amount of missing values in the data set and since it returned a small quantity, representing only 2% of all rows, the missing values were imputed as the mean-value of their respective column. The imputation of the missing values with the mean was sufficient in this data set, due to the non-significant changes in the variance.
The task was divided into two phases. In the first phase, it was required that the classifiers were trained based on the categorical data "goal", which could vary from presence (values 1,2,3,4) to no presence (value 0). In the second phase, the categorical data "goal" was transformed into a binary outcome "no heart disease vs. heart disease" (0 and 1) and each classifier was trained once more based on the new class labels.

4 Methods

For the purpose of exploratory analysis, descriptive functions for data frame, pair plots and heatmaps were used in order to have first insights of the data, understanding each variable and studying possible correlations between the attributes.
Then, in order to develop three classifiers to predict CAD and subsequently evaluate them, the data set was split into 80% training and 20% test set, using the inbuilt function `.sample()` of the `random` module in Python. Using the machine learning library `scikit-learn`[2], three basic classification solutions were implemented: Logistic Regression, K-Nearest Neighbor and Decision trees. All classifiers were trained using the same training set. For evaluation and comparison purpose,

[1]https://archive.ics.uci.edu/ml/datasets/Heart+Disease

a Confusion Matrix, Classification Report and ROC Curves were generated using the library `scikit-learn`.

5 Results

In the first phase of the task, it was observed that the data was unbalanced, since the amount of ”no presence” values was significantly higher than the other categories referred to the presence of heart disease. This discrepancy can be seen in Figure 1.

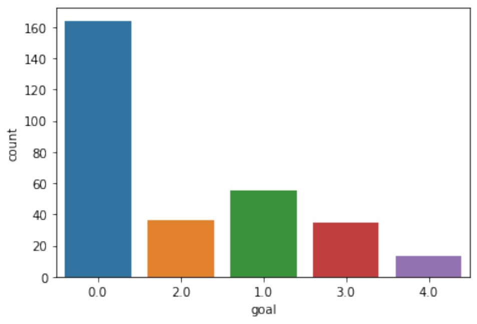

Figure 1 Imbalanced classification in the Phase 1

The classifiers were first trained on the imbalanced data and the results can be seen in 2. In terms of accuracy, the Decision Tree Classifier presented a train accuracy of 100%, which leads to the inference that this prediction model was overtrained because the test accuracy is much lower at 60,7%. The Logistic Regression and the K-nearest neighbor showed a similar result, with an accuracy of 64,9% (train) and 62,3% (test) and 67,4% and 57,4% (test), respectively. Considering the precision and recall outcome, the classifiers showed a similar pattern, with K-nearest Neighbor in the last position and the classifiers Logistic Regression and Decision Tree with a close performance.

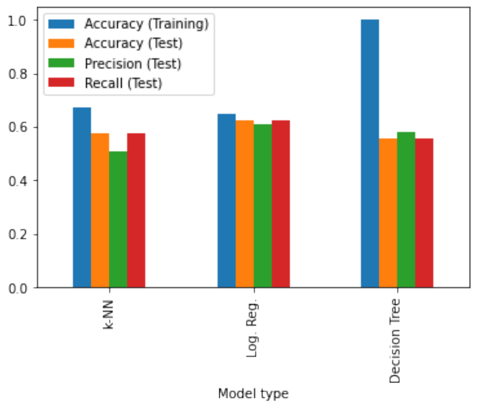

Figure 2 Performance of classifiers in Phase 1

In the second phase of the task, the data set was balanced, as the minority classes 1, 2, 3 and 4 were merged into only one class describing the presence of heart disease. Therefore, the category ”goal” was converted into a binary class, in which values could take 1 for the presence of heart disease or 0 for no presence. The new classification can be seen in Figure 3.

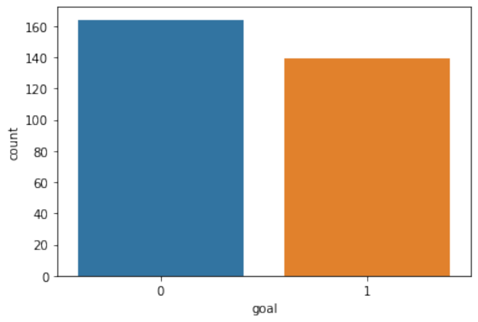

Figure 3 Balanced classification in Phase 2

The classifiers were trained based on the new target group, the results can be found in Figure 4. The classifiers perform better compared to the previous phase. The result of the accuracy during the training for the Decision Tree suggests that the model again overfitted the data, while K-nearest Neighbor and Logistic Regression presented accuracies of 83.9% and 84.7% during the training.

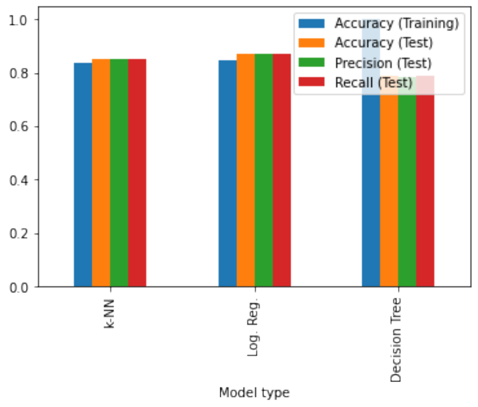

Figure 4 Performance of classifiers in the Phase 2

With regard to the test data set, the Logistic Regression provided the best accuracy around 86.9%, followed by K-nearest Neighbor with 85.2% and Decision Tree with 81.2%. In terms of precision and recall, the Logistic Regression presented a level of 87% and 87%, respectively, against 85% precision and 85% recall from K-nearest Neighbor and 82% precision and 82% recall from Decision Tree.
To evaluate and compare the performance of the classifiers, a ROC Curve was generated, which can be seen in Figures 5, 6 and 7. Comparing the ROC Curves, it is observed that the Logistic Regression performed better in relation to the other

classifiers, since it presented a curve closer to the upper left corner of the graph, indicating that the true positive rate is higher compared to the false positive rate.

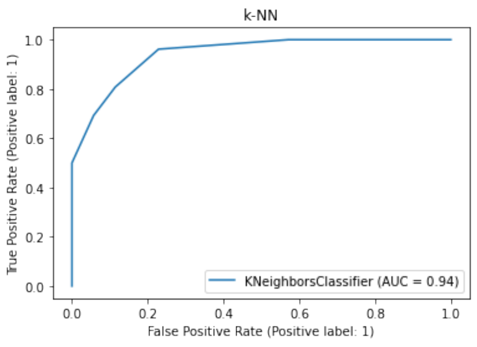

Figure 5 ROC Curve from the K-nearest Neighbor Classifier

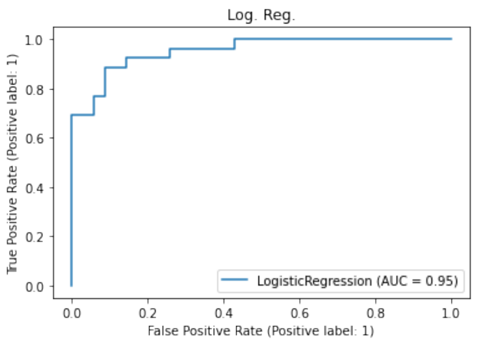

Figure 6 ROC Curve from the Logistic Regression Classifier

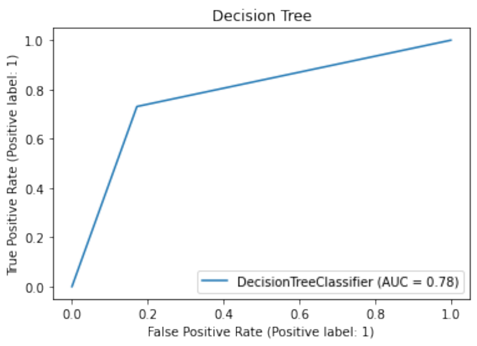

Figure 7 ROC Curve from the Decision Tree Classifier

6 Discussion

After comparing the results of the classifiers in phase 1 and 2 of the project, it was evident to us how the balance of the data is affecting the performance of the prediction models. When training with balanced data, the classifiers could better predict the presence of heart disease. The best prediction model applied to this dataset was the Logistic Regression Classifier, which showed an accuracy of about 87%.
During the exploratory analysis, we could detect some problems with the data, such as missing values and imbalanced data. The first was solved by imputation with the respective mean, a method introduced in the lectures. The problem with imbalanced data was overcome by merging the minority classes into one class, which converted the target group into a binary class. In the implementation of the classifiers, the Decision Tree based prediction model was overtrained. The solution to this problem was not explored in practice due to time constraints, but two different techniques could be implemented: Pre-pruning and Post-pruning. Pre-pruning is stopping the growth of the decision tree earlier, while post-pruning is generating a complete tree and later removing some of the branches. These two techniques would prevent overtraining when implementing a decision tree classifier.
This project provided us with good knowledge of a typical project for a data scientist. In particular, in dealing with data preprocessing and problems such as unbalanced data and missing values, typical issues in data science were encountered. In addition, by participating in this project, we were able to develop practical skills with regard to machine learning and the implementation of classifiers.

7 Appendix

Workload distribution:

Dominik Bannwitz (Bioinformatics) - contributed 16 hours
- report: scientific background, results

Florian Herzler (Bioinformatics) - contributed 16 hours
- report: data + preprocessing, methods

Maximilian Otto (Bioinformatics) - contributed 16 hours
- code: data, classification

Mariana Steffens (Data Science) - contributed 6 hours
- code: fruit example
- report: scientific background

References
1. Detrano, R., Janosi, A., Steinbrunn, W., Pfisterer, M., Schmid, J., Sandhu, S., Guppy, K.H., Lee, S., Froelicher, V.: International application of a new probability algorithm for the diagnosis of coronary artery disease. The American Journal of Cardiology **64**, 304–310 (1989)
2. Pedregosa, F., Varoquaux, G., Gramfort, A., Michel, V., Thirion, B., Grisel, O., Blondel, M., Prettenhofer, P., Weiss, R., Dubourg, V., Vanderplas, J., Passos, A., Cournapeau, D., Brucher, M., Perrot, M., Duchesnay, E.: Scikit-learn: Machine learning in Python. Journal of Machine Learning Research **12**, 2825–2830 (2011)

RESEARCH

Introduction to Focus Areas - Project 2 - Group 8

Maximilian Otto[*], Dominik Bannwitz, Mariana Steffens and Florian Herzler

[*]Correspondence:
maximilian.otto@fu-berlin.de
Department of Mathematics and Computer Science, Free University of Berlin, Takustraße 9, 14195 Berlin, Germany
Full list of author information is available at the end of the article
[†]Equal contributor

Abstract

Goal of the project: Classification of images into benign or malignant.

Methods used in the project: 3 different residual neural networks in PyTorch.

Main results of the project: ResNet18 performed better than the more complicated models.

Personal key learnings: Implementation of residual neural networks in PyTorch.

Estimated working hours: 54

Possible improvements: Fine tuning of hyperparameters, downsampling to balance classes, further pre-processing of images

Keywords: Classifiers; Deep Learning; Breast Cancer; ResNet; ResNeXt

1 Scientific Background

Classification in data science is used to classify data into classes, in our case classifying images of tissue samples into benign or malignant tumor classes. A trained classification model can be used, to predict the classes of new, similar data. The used data is unstructured, as it is not stored in tabular but in image form. The information contained in the images can not be accessed directly, but has to be extracted by more complex methods. Hence, the data needs to be split in a train and test set to train a prediction model and evaluate their overall performance. Also, the data can be transformed beforehand to artificially obtain more training data.

2 Goal

The goal of the authors was to provide a publicly available data set of breast cancer histopathology images and perform automated classification into benign or malignant images as a supporting tool for pathologists. Similar to the paper [1] by *Spanhol et al. 2015*, we were tasked with developing and evaluating three different Deep Learning (DL) topologies that work directly on the images of the provided data set. Our goal was to correctly classify as many images of the labeled *test* data as possible as one of the binary outcomes: benign (B) or malignant (M).

3 Data and Preprocessing

Data

The provided publicly available data set[1] consists of 7909 images of breast tissue of 82 patients, classified into different classes of benign (*adenosis, fibroadenoma, phyllodes tumor, tubular adenoma*) and malignant (*ductal carcinoma, lobular carcinoma, mucinous carcinoma, papillary carcinoma*) tumors. The images were obtained in surgical open biopsy (SOB) and classified and labeled as one of the above tumor types by a pathologist. For each type, *Spanhol et al.* provide different microscopic magnifying factors of the tissues (Table 1).

Magnification	Benign	Malignant	Total
40X	652	1,370	1,995
100X	644	1,437	2,081
200X	623	1,390	2,013
400X	588	1,232	1,820
Total of Images	2,480	5,429	7,909

Table 1 Distribution and amounts of breast cancer images with different magnification factors.

We focus our work on just one subset of the data (*fold2*)[2].

We chose magnification factor 200X and thus ended up with all of the images of factor 200X re-ordered in the corresponding benign or malignant sub folder in a 70 to 30 percent split between training and testing data, similar to *Spanhol et al.*

Preprocessing

Spanhol et al. also provide a Python script [3] to distribute the images to training and testing folders. Because the authors ran five trials, we modified this script to only re-order the images once, creating a **train** and a **test** directory with sub directories **B** and **M** (benign and malignant) for one magnifying factor. No changes were made to the original data, as there were no missing values in the image data set and we did not balance the sample size of the classes.

Several data augmentations were performed, which got executed during the runtime, but could also be associated as a part of the preprocessing. After downsizing the images to $224x224$ pixels, we used randomized horizontal and vertical flips, a random rotation of each image of $(-20, 20)$ degrees, and normalized the images by the mean and standard deviation of all pixels within the train set. The data augmentations are used to simulate more data than provided to tackle an immediate overtraining and give the model a chance to learn more generalized features.

[1] http://www.inf.ufpr.br/vri/databases/BreaKHis_v1.tar.gz
[2] www.inf.ufpr.br/lesoliveira/download/TBME-00608-2015-R2-preprint.pdf
[3] http://www.inf.ufpr.br/lesoliveira/download/mkfold.tar.gz

4 Methods

The `Python` library of choice was `PyTorch` [2]. From that library three different networks were used: ResNet-18 [3], wide-ResNet-50[4] and ResNeXt-50_32xd4[5]. All three are residual neural networks, which provide an architecture similar to classical Convolutional Neural Networks (CNN). ResNet-networks usually have between 18 and 152 convolutional layers, but can support up to thousands of layers. Shallow CNNs with that many layers would run into the "vanishing gradient problem", which occurs when the algorithm tries to find small weights and by that the loss function is set to the minimal value. With too many layers this leads to a very small gradient until it disappears and no further activation of the following neurons can be registered. A residual network solves this problem by adding *skip-connections*, which add a residue of the previously executed block to the output of a current residual block. Each block consists of multiple convolutional layers, followed by activation functions and batch-normalizations. By using this method, they tackle the "vanishing gradient problem" and can outperform traditional CNNs. That is why we have chosen residual networks as our preferred, underlying architecture.
Wide-ResNet und ResNeXt are variations of residual networks, which come with their own advantages and disadvantages. ResNeXt repeats a build-block of ResNet. This enables the ResNeXt to split the input of each block into different channels and analyze spatial relationships, due to the individual pathways within each block. Wide-ResNets simply add more channels to the ResNet-architecture and transform the data into even higher dimensions. Thereby, our Wide-ResNet-50 had about 60 million trainable parameters compared to 25 million of the used ResNeXt-50. The main hyperparameters for this networks remained the same for all models: number of classes, learning rate, the batch size, drop out parameter, and optimizer (Table 2).

Parameter	f*Value*	*Short description*
Number of epoch	5	How many epochs the network runs through
Number of classes	2	How many output classes to classify
Learning rate	0.0005	Rate with which the weights are modulated
Batch size	32	Image per iteration
Drop out rate	0.4	Probability of eliminating a neuron
Optimizer	ADAMAX	Gradient descent method

Table 2 Main hyperparameters for training our neural networks

We used the same testing to training split ratio as *Spanhol et al.* with 70% training data to 30% testing data. We tested different combinations of hyperparameters and modified the learning rate with an exponential learning rate scheduler throughout training. The ResNet-18 got trained for 15 epochs. To shorten the training time, we used pre-trained networks. Thy were previously trained on *ImageNet* [6] and we re-trained all layers. The fully connected layer for classification got replaced with three fully connected layers (# outputs: 256, 100, 2) and *ReLU* as an activation function and a drop-out layer in between.
The evaluation consisted of displaying the precision and recall for both classes B and M in a confusion matrix. Additionally, we created heatmaps of the confusion matrices. We also kept track of accuracy and loss (*CrossEntropyLoss*) for both training and test sets and validated the results with the weighted *F1-Score*.

5 Results

To evaluate the three residual neural networks precision and recall as well as the *F1-Score* were determined. Figure 1 shows the confusion matrices for the predictions of the three residual networks on the test data. From these it can be recognised, that the ResNet18 correctly predicted the most malignant tumors on the images with an amount of 404 out of 429 in the test data.
ResNext50 correctly classified only a bit less then ResNet18 with 393, but Wide-RestNet50 was only able to predict 295 of the malignant tumors out of the 429 correctly.
For the prediction of the benign data, it can be recognised in Figure 1 that this was less successful than the classification of malignant tissue data. The RestNet18 was only able to predict 119 out of 168 of the benign tumors on the pictures correctly. The ResNext50 network was again a little less successful than the ResNet18 by predicting 115 of the benign tumors, but the Wide-ResNet50 outperformed the two other networks by correctly predicting 138 of the 168 benign tumors.

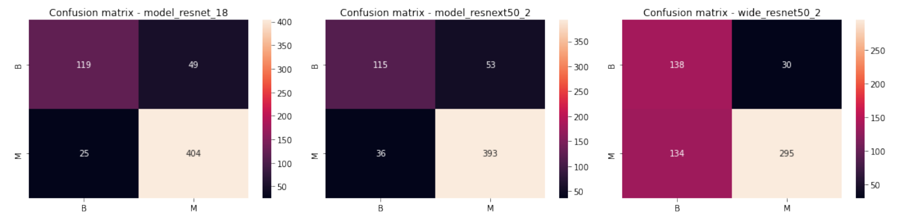

Figure 1 Confusion matrices

ResNet-18

The ResNet18 network reached the highest *F1-Score* out of the three trained models with 87% accuracy. The precision on the benign data lays at 83% and the recall at 71%. The malignant data could be predicted with a precision of 89% and a recall of 94% (See table 3).

	precision	recall	f1-score	support
B	0.83	0.71	0.76	168
M	0.89	0.94	0.92	429
weighted avg	0.87	0.88	**0.87**	597

Table 3 Precision, recall and *F1-Score* of the ResNet18 network

ResNeXt-50

With a f1-score of 85% the ResNeXt-50 network had the second highest score. When predicting the benign data, it only reached a precision of 76% and a recall of 68%. The prediction for the malignant data works a lot better with a precision of 88% and recall of 92% (see Table 4 on the facing page).

	precision	recall	f1-score	support
B	0.76	0.68	0.72	168
M	0.88	0.92	0.90	429
weighted avg	0.85	0.85	**0.85**	597

Table 4 Precision, recall and *F1-Score* of the ResNeXt50 network

Wide ResNet-50

The wide ResNet-50 reached only a *F1-Score* of 74% and performed the worst out of our three residual networks. The prediction for the benign tissue images reached only a precision of 51% and a recall of 82%. For the malignant it looks mirrored. The prediction for malignant tissue samples yields a precision of 91%, which is the highest one of all tree networks, but only a recall of 69% (see Table 5).

	precision	recall	f1-score	support
B	0.51	0.82	0.63	168
M	0.91	0.69	0.78	429
weighted avg	0.80	0.73	**0.74**	597

Table 5 Precision, recall and *F1-Score* of the Wide ResNet50 network

6 Discussion

Analyzing the performance of the three residual models the first thing what can be compared is the accuracy and loss function on the training vs. the test data over the different epochs. The ResNeXt-50 and wide ResNet-18 are both getting a higher accuracy and a smaller loss on the training data then on the test data (see appendix figure 3 on the following page & figure 4 on the next page). This is a sign of overtraining, which fits in well with the fact that the two networks are significantly more complicated than ResNet-18. The latter also seems to overtrain in some epochs, but on the other hand the performance on the test data was even better then on the training data (see Figure 2). It seems that simpler model architectures suffice.

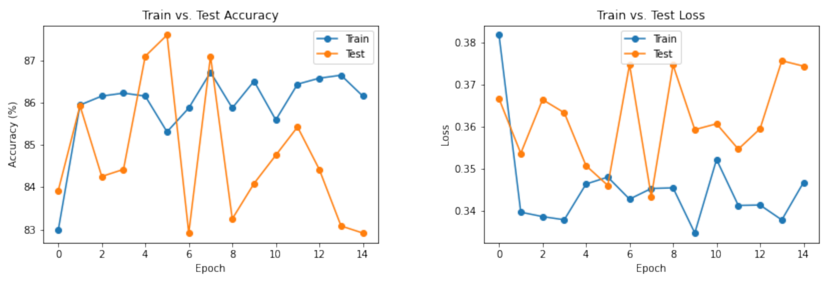

Figure 2 ResNet-18: Accuracy and loss

Improvements

The original data set contains pictures with four different magnifying factors. Since the 200X images performed best in the original paper we also used them to train our networks. On way to overcome the over training problem could be to use instead all the pictures. Another ratio of training and test data set size (80 / 20 split) could also lead to more stable training results.
Fixing the class imbalance problem would decrease the amount of overtraining for only one class, because in the training phase the model would see the same number of images of benign and malignant tissue.
Fine tuning the hyperparameters can always help to improve the performance of a model. Our learning rate is quite small, but tests with 0.1 instead of 0.005 produced a worse performance. We would suggest to further tune the parameters. Batch size and number of epochs can also be modified, but these might not lead to an improved performance.

7 Appendix

Figures

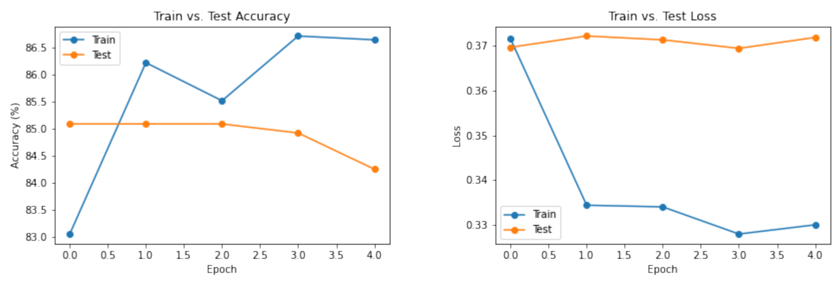

Figure 3 ResNeXt-50: Accuracy and loss

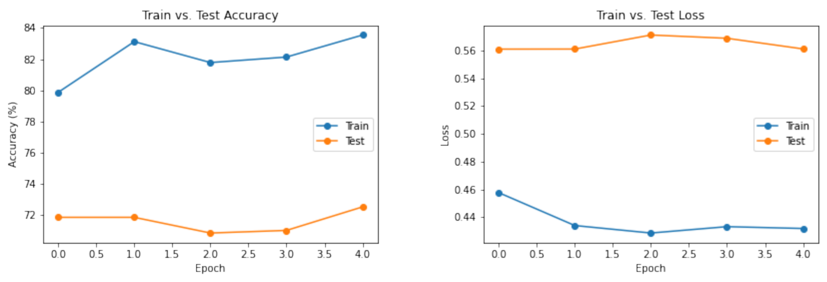

Figure 4 Wide ResNet-50: Accuracy and loss

Workload distribution:

Dominik Bannwitz (Bioinformatics) - contributed 16 hours
- report: scientific background, results, discussion

Florian Herzler (Bioinformatics) - contributed 16 hours
- code: data + preprocessing
- report: data + preprocessing, methods, improvements

Maximilian Otto (Bioinformatics) - contributed 16 hours
- code: models + evaluation
- report: methods

Mariana Steffens (Data Science) - contributed 6 hours
- report: goal, discussion

References

1. 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. Accessed 2022-11-18
2. Paszke, A., Gross, S., Massa, F., Lerer, A., Bradbury, J., Chanan, G., Killeen, T., Lin, Z., Gimelshein, N., Antiga, L., Desmaison, A., Köpf, A., Yang, E.Z., DeVito, Z., Raison, M., Tejani, A., Chilamkurthy, S., Steiner, B., Fang, L., Bai, J., Chintala, S.: Pytorch: An imperative style, high-performance deep learning library. CoRR **abs/1912.01703** (2019). 1912.01703
3. He, K., Zhang, X., Ren, S., Sun, J.: Deep Residual Learning for Image Recognition. arXiv. Number: arXiv:1512.03385 arXiv:1512.03385 [cs] version: 1 (2015). doi:10.48550/arXiv.1512.03385. `http://arxiv.org/abs/1512.03385` Accessed 2022-11-18
4. Zagoruyko, S., Komodakis, N.: Wide Residual Networks. arXiv. Number: arXiv:1605.07146 arXiv:1605.07146 [cs] version: 4 (2017). doi:10.48550/arXiv.1605.07146. `http://arxiv.org/abs/1605.07146` Accessed 2022-11-18
5. Xie, S., Girshick, R., Dollár, P., Tu, Z., He, K.: Aggregated Residual Transformations for Deep Neural Networks. arXiv. Number: arXiv:1611.05431 arXiv:1611.05431 [cs] version: 2 (2017). doi:10.48550/arXiv.1611.05431. `http://arxiv.org/abs/1611.05431` Accessed 2022-11-18
6. Deng, J., Dong, W., Socher, R., Li, L.-J., Li, K., Fei-Fei, L.: Imagenet: A large-scale hierarchical image database. In: 2009 IEEE Conference on Computer Vision and Pattern Recognition, pp. 248–255 (2009). doi:10.1109/CVPR.2009.5206848