Development of an end-to-end web application for visualization, evaluation, and post-processing of result data from neural network predictions for the melanoma use case

Chiara Tappermann*[1], Mohan Xu*[1][0000−0001−6940−1361], Lena Wiese[1,2][0000−0003−3515−9209], and Babak Saremi[1][0009−0008−0269−4962]

[1] Fraunhofer Institute for Toxicology and Experimental Medicine, Hannover, Germany
[2] Institute of Computer Science, Goethe University Frankfurt, Frankfurt a. M., Germany
{chiara.tappermann|mohan.xu|lena.wiese|babak.saremi}@item.fraunhofer.de

Abstract. Image acquisition technology advances in various fields, yet current image analysis tools limit the effective application of image analysis due to their cumbersome operation process and the requirement of professional knowledge and skills. In this paper, we develop a semi-automated web application for image segmentation and classification tasks with the support of neural networks (relieving the above-mentioned current research dilemma) using melanoma detection as a use case. The web application enables scientists to participate and improve the decision-making process of the neural network through the concept of "human-in-the-loop", while saving expensive labor costs due to its automation in image annotation and classification. In addition, our web application achieves high usability in the general user community by testing seven aspects of usability: The first impression, distingishability and clarity of the tools, intuitive characteristics, learnability, feedback and reaction, implementation of expected functionality, and the fullfillment of usability.

1 Introduction

Web applications have become an essential part of our daily lives, enabling us to perform many tasks through a web browser. They have a lot of benefits compared to conventional desktop software. In the context of our work, web applications are software that can be accessed through a web browser and require an active internet connection. Web applications can provide the user with powerful tools without the need for a high-performance infrastructure from the user side because the backend can be deployed on external powerful servers. In addition, the increasing popularity and power of neural networks shaped many different

* Equal contribution as first authors.

fields of science. Neural networks are able to process numerous tasks on huge data sets to extract important features. One popular application area is the image analysis of biomedical data [18, 4, 24, 17, 15]. With the increasing amount of image data produced, scientists are in need of specific tools that can help to process these data sets. This process usually requires expert knowledge, especially in the case of biomedical image data where complex cellular structures need to be evaluated. Applications such as ImageJ [31] and Labelme [37] are able to select a region of interest in a series of images and evaluate that region. However, these evaluation processes are done entirely by hand and can therefore be a very time-consuming task. A well-trained neural network can support scientists by automatizing several processes and thus save manpower. In addition, it was shown in the past that neural networks can sometimes work more accurately than humans and thus consequently reduce human error rates [2, 38, 14]. However, for some complex cases, neural networks are unable to predict desired features correctly, which can often be the cause of insufficient training data availability [30]. Another important issue is the acceptance of neural networks in medical sectors, as these models are often considered a black box and misclassifications can lead to further distrust in AI-generated results [32].
For our purposes of melanoma detection, we developed a web application that facilitates access to neural networks that are provided in the backend of our application to perform image predictions. To address the above-mentioned issues, we present a fully functional prototype for image segmentation and classification tasks that use the "human-in-the-loop" concept [4]. This approach combines the power and accuracy of a neural network and the expert knowledge of a scientist who can intervene and correct the classifications done by a neural network and thus help to annotate and evaluate image series more accurately and enable the scientists to perform their evaluation process using a partially automated pipeline instead of their currently completely manual process.
In this paper, we extend our prior work [36, 39] for our prototype and focus on the identified users' requirements. We also elaborate on the target platform and library chosen for the implementation of the application. Another aspect we focused on is the software quality of the application. We analyzed the requirements, implemented and tested the software with a focus on usability, and also the possibility to extend the software or adjust it for further use cases.

2 Related Work

In recent years, image analysis tools have received a lot of attention from the community due to the increasing amount of image data and their versatility in different categories of images. ImageJ [31] and Labelme [37], two of the most commonly used image editing tools, can perform a series of post-processing operations such as modifying and analyzing the annotated images while completing image annotation on the dataset. These operations are done manually, which not only reduces the efficiency of scientists' work but also increases the potential human error rate. In response, several works based on machine learning algorithms

designed image analysis tools to make image analysis equally efficient in large datasets. As a plugin for ImageJ, [1] implements a continuously trainable pixel classification workflow. Based on similar concept, the interactive tool proposed in [3] covers several workflows such as image segmentation, object classification, counting, and tracking. However, the operation of these two interactive tools not only requires the computing power of local devices, but also requires debugging the local environment during installation. The web application developed in [23, 21] is not limited by local devices and implements the functions of region size quantification, cell counting, object classification and interest points detection. Although current commercially available image analysis tools are rich in features, the redundant features rather increase the difficulty for scientists to get started. Moreover, the machine learning models they train and the accompanying image analysis methods are not suitable for all use cases. Therefore, it is necessary to develop an end-to-end trainable web application for these use cases.

3 Fundamentals

3.1 Melanoma use case

According to the American Cancer Society [33], there were 97,920 new cases of melanoma in situ of the skin in the United States in 2022, including 7,650 deaths. If melanoma is detected and treated promptly in early diagnosis, the 5-year relative survival rate can reach 93%, second only to prostate cancer (98%). However, dermoscopy, an important tool for the early detection of melanoma, is highly dependent on the clinical experience of dermatologists [10]. Therefore, the development of a tool that encompasses the clinical experience of the physician and the analytical power of neural networks to segment and classify lesion areas has a positive effect on improving the early diagnosis of melanoma.

3.2 Concept of the application

Our goal was to develop a partially automated tool for image analysis that is supported by convolutional neural networks (CNNs) which can detect the shape of a skin lesion in images and classify the lesion for the melanoma use case. To make this technology available to a variety of end-users, we focused on implementing an intuitive, web-based end-to-end-pipeline, that has high usability and offers services to segment and classify skin lesions, to post-process predictions and perform several evaluation methods on the data based on the user's needs. The main advantage of our service is, that it can improve the CNNs iteratively with an interactive "human-in-the-loop" approach; hence, our workflow provides the user with constantly improving technology while expanding our dataset to make this technology more accurate. Due to our modular architecture, our service can be run on-premise to avoid privacy implication.

3.3 Tasks and actors

There are two types of actors or users for the software system participating in the workflow shown in figure 1. We have scientists or physicians as primary actors. They interact with the application directly through the user interface and process their data through it. The secondary actor is the software developer, whose job is to improve the prediction results by providing the user with updated versions of the neural networks in the background for the data prediction. Based on the different actors, we identified the tasks for the application by conducting interviews with potential end-users and by analyzing the data format. Identifying the tasks is an important aspect of the software engineering process to describe the interaction between an agent and a software system. It can be used to specify the requirements for the application [13, 34]. As a result, we extracted the following tasks for the primary actor:

1. **Process files:** Upload image data or progress files
2. **Process files:** Download image data or progress files
3. **Process files:** Generate prediction for image data
4. **Process files:** Export the generated annotations for retraining
5. **Image processing:** Post-Process or correct predictions
6. **Analysis:** Apply different operations to generate additional data for analysis
7. **Analysis:** Save the processed data in a suitable format
8. **Analysis:** Edit the processed data
9. **Visualization:** Visualize the processed data for the user
10. **Visualization:** Export the results
11. **Additional Features:** Tutorials on how to use the application

The tasks for the secondary actor are the following:

1. **Data generation:** Create labeled data
2. **Neural network:** (Re-)train the neural network
3. **Neural network:** Publish the network for the user

Based on these tasks, our result is an end-to-end pipeline that accesses the CNNs in the backend and a web application in the frontend. In particular, the developer is primarily responsible for training the neural networks using the provided labeled data, which can be provided either by the developer or the user depending on their domain knowledge. Labeled data can easily be created using the editor tool provided by our web service. Users can apply our application to their data to correct inaccurate predictions. The corrected data can be used to extend the dataset and improve the neural network's accuracy through iterative retraining. This process ultimately reduces the workload for users, as fewer post-processing operations are required when the neural network's performance improves. Users can then upload either a single image or image series, depending on the application, to generate a segmentation of the lesion area and a classification if the lesion area is a melanoma or not. Then they are able to correct

faulty predictions for either single images or image series. For image series, users can keep previous changes without starting the correction process over again for each image. Additionally, users can perform further operations such as calculating the size of the melanoma, to detect changes in size over time for different images of the same melanoma. Users can input metadata and get an overview of all previously created data. In this step, the users can also export the image data and corresponding labels to expand the dataset. In the final step of the workflow, users can visualize the data and export the results. For the melanoma example, we visualized patient records containing images of the patient's melanoma from different examination dates. The files also provide the users with information about the count of melanoma detected for each examination and a classification for each image of the mole, classifying if it is a melanoma or not.

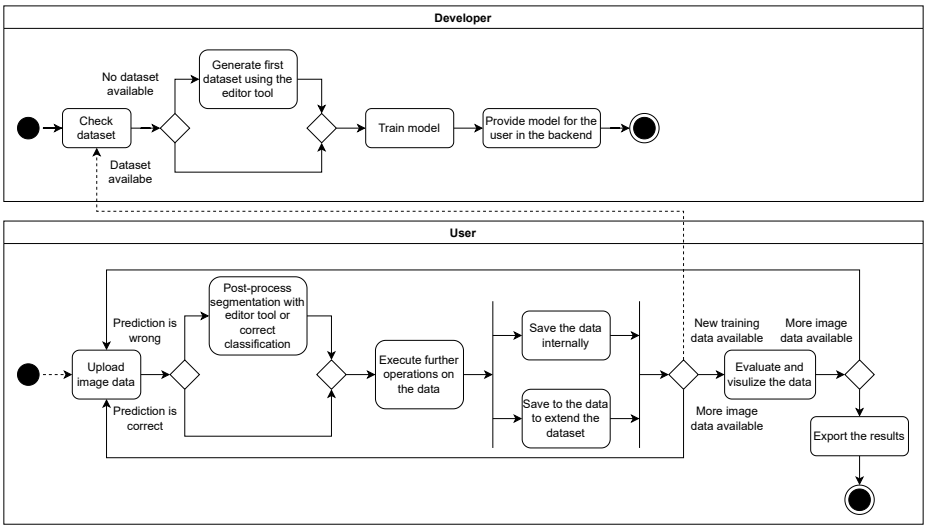


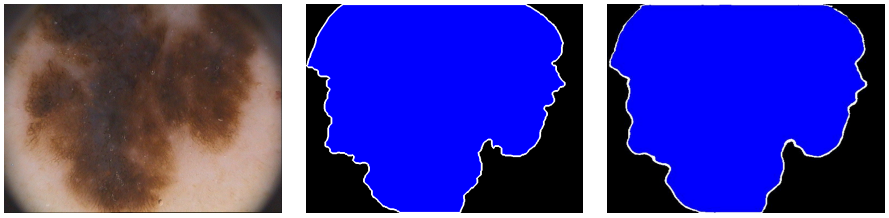
Fig. 1. Workflow of the developer and the user

4 Material and methods

4.1 Dataset

We completed the training and testing of our neural network for image classification and segmentation on the publicly available PH2 [25] dataset, which contains 200 dermoscopic images. The dataset was partitioned for the image segmentation and the image classification task. For the image classification task, the dataset contains 160 nevis and 40 melanomas and the corresponding labels, where 0 indicates a nevus, and 1 indicates a melanoma. The dataset was partitioned into

a training- and a test set in a ratio of 7:3. In contrast, the image segmentation task dataset contains 200 images with their corresponding annotations, where a pixel value of 0 represented the background and a pixel value of 1 represents the region of interest. The dataset was again partitioned into a train and a test set in the ratio of 7:3. An example for a melanoma image and the corresponding segmentation ground truth and also the prediction is provided in figure 2.



(a) (b) (c)

Fig. 2. Image data 2(a), ground truth 2(b), prediction 2(c)

4.2 Classification and segmentation with neural networks

For the image classification task, we used the classical neural network resnet50 [16] to determine whether an object from the PH2 dataset is lesioned. The neural network consists of 5 convolutional structures, 1 mean pooling layer and 1 fully connected layer that outputs the two predicted classes 0 for nevi and 1 for melanoma. Considering the small PH2 dataset, we used weights pre-trained on Imagenet [9] in order to make the model converge better and faster. For the image segmentation task, we used nine different neural networks defined by segmentation models from Pytorch[3] which are all popular architectures for image segmentation: Unet [29], Unet++ [41], Deeplabv3 [6], Deeplabv3+ [7], FPN [20], Linknet [5], PSPNet [40], PAN [19] and MAnet [11]. The model architectures consist of an encoder, a decoder, and a segmentation head. The encoder ("backbone network") is responsible for feature extraction. The decoder is responsible for feature fusion and progressive implementation of category annotations for each pixel. The segmentation head ensures that the number of channels from decoder output is equal to the number of prediction categories while ensuring that the size of input and output are the same by up-sampling.

4.3 Frontend technologies

We implement the frontend and user interface as a web application using the framework Angular, an open-source JavaScript framework, based on TypeScript.

[3] Iakubovskii, P.: `https://github.com/qubvel/segmentation_models.pytorch`

An Angular application consists of modules that again are composed of components developed and published by Google. This modular architecture helps to structure large-scale software projects while keeping the application maintenance extendable at a later time [22]. A simple, well-structured and consistent design is important for a good user interface. Therefore, we used the component-based design library *Material Design*[4] that provides us with a broad range of interactive components that can be utilized across multiple different platforms. The Material Design library was developed by Google in accordance with the best practice guidelines and is therefore compatible with Angular applications. In order to develop an editor for the post-processing stage of segmenting results, we had to decide on a method for image data processing. There are different ways to display and edit image data, including raster images and vector graphics. Vector graphics are defined using objects like paths and shapes, where the information about those objects is described through mathematical formulas [12]. We analyzed different vector-based libraries for the implementation of our application and decided to use Paper.js[5]. Paper.js provides a wide range of basic vector-based operations and allows us to implement all required tools.

4.4 Backend technologies

MLOps is a standard set of working procedures designed to manage all life cycles of machine learning models. It implements a set of operations for continuous integration, continuous delivery, and continuous deployment of machine learning models in a production environment. In our use case, we used *TorchServe*[6], developed specifically for the *Pytorch*[7] framework, to deploy the image classification model and image segmentation model mentioned in section 4.2. After receiving a prediction task request (classification or segmentation) and data from the web application, the handler component of TorchServe will process and predict the data corresponding to the task request and return the predicted output to the web application. To ensure a consistent run time environment and to enable fast deployment of models, we install and run TorchServe within docker. Figure 3 shows the architecture of our web application and illustrates the communication between the frontend and the backend. The frontend communicates with the backend through a REST API interface. The backend consists of either *TFServing*[8] [26] or TorchServe[9] backend and is responsible for deploying trained image classification and segmentation models, assigning workers to the deployed models and communicating with the frontend. After processing the REST requests from the frontend, the backend generates a JSON file, with the generated prediction.

[4] Material Design, `https://material.io/`
[5] Paper.js. `http://paperjs.org/`
[6] TorchServe, `https://pytorch.org/serve/`
[7] Pytorch, `https://pytorch.org/`
[8] TFServing, `https://www.tensorflow.org/tfx/guide/serving`
[9] Torchserve. `https://pytorch.org/serve/`

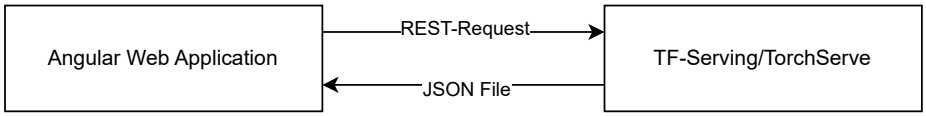


Fig. 3. Communication between the web application frontend (left) and backend (right)

5 Experiment

In our work, we focused on the usability of our web application. Therefore, we performed a study testing our labeling tool to evaluate the usability of our user interface. The topic of usability covers a broad range of different aspects of user interface design and workflow design in the software engineering process. Usability describes the relationship between the user, the software, and a task to be performed and provides a measure to determine the quality of the tool to solve the task considering aspects like efficiency, understandability or intuitivity [28]. The goal of this study was to assess the intuitiveness of the application and the user's ability to use the tools in a meaningful manner for a specific use case. Our research hypotheses for the study were:

1. The application contains all the tools needed to fulfill the defined use cases
2. The application is intuitively structured to ensure that the user quickly finds his way for efficient use of the required tools

The study was conducted with a senior scientist as an end user, who had to edit data for a comparable use case, without prior explanation of the application. During the study, several questionnaires were provided, which included questions for both quantitative and qualitative evaluation. Quantitative evaluation methods allow for a precise statistical evaluation and have the advantage to uncover sources of errors. Although these methods are more complicated, they can give suggestions for improvement. Accordingly, in a study where few participants are expected, it is reasonable to also use qualitative methods in order to obtain more precise findings [35]. The questionnaires are based on the questionnaire developed by Jochen Prümper and Michael Anft in 1993, which is based on the international ergonomics standard ISO 9241/10 [35, 27]. Parts of the questionnaire were adjusted to the specific use case and categorized into the following sections defining seven different aspects of usability: The first impression, distinguishability and clarity of the tools, intuitive characteristics, learnability, feedback and reaction, implementation of expected functionality, and the fulfillment of usability. These categories were derived from Chlebek's usability criteria [8]. The study was separated into six different sections. At the beginning of the study, the participant was informed about the process and the objectives of the study, followed by demographic questions. In the third stage of the study, the participant was allowed to test the application briefly and ask questions. Following the familiarization period, a short questionnaire was answered to obtain the first impressions. The fourth step was to work on a suitable use case, using the

labeling tool. During this familiarization period, the participant commented on the process and reported on anything that was noticed. Afterward, the remaining questionnaire was filled out, which dealt in detail with different categories of usability. In the last section of the study, a follow-up discussion of the study was conducted with the participant, who was able to discuss impressions of the study and the application. In summary, the outcome of the study was very successful. The individual aspects of the application were rated good to very good in the questionnaires. Some minor improvements were suggested that have already been implemented in the current version of the application. Therefore the research hypothesis of the usability study can be answered positively.

6 Results

The result of our implementation process is an Angular web application, that communicates with TFServing or Torchserve in the backend to generate a classification and segmentation mask for skin lesions for the melanoma use case. The application has a modular design and consists of a number of components and services that create the structure of the start page the image editor page for the data labeling process, the uploader page to process and evaluate image data for the data visualization, the data visualization page, the tutorial page, and the page to export the data generated during the evaluation process to expand the training dataset. All subpages can be accessed from the start page. The data uploader illustrated in figure 4 allows the user to upload images or image series and either further process the segmentation mask or adjust the classification result and also perform further calculations on the segmentation mask. The editor page is illustrated in figure 5 and can be used to either generate new labels for the training dataset or adjusts the segmentation mask. It was designed based on other common image-processing tools where users can select the corresponding tool in the toolbar located on the left side of the screen, and each tool corresponds to several options, which can be found in the image bar above. On the right are further options such as layer- or saving options. Figure 6 also shows an example of how the data can be visualized as patient records, showing all the patient's documented skin lesions, the corresponding classification, and how many melanomas were found over time.

7 Discussion

For the user interface of our application, we decided to use a Web application. This type of software can be used on a number of different devices that supports modern web browsers, regardless of the underlying operating system. Since the service is hosted on an external server that performs all necessary computations, web apps are not restricted by the computational capability of the user's local device. Web applications offer many benefits, but they also have disadvantages. For instance, the user must be connected to the Internet at all times, and delays can occur when dealing with large amounts of data if the Internet connection

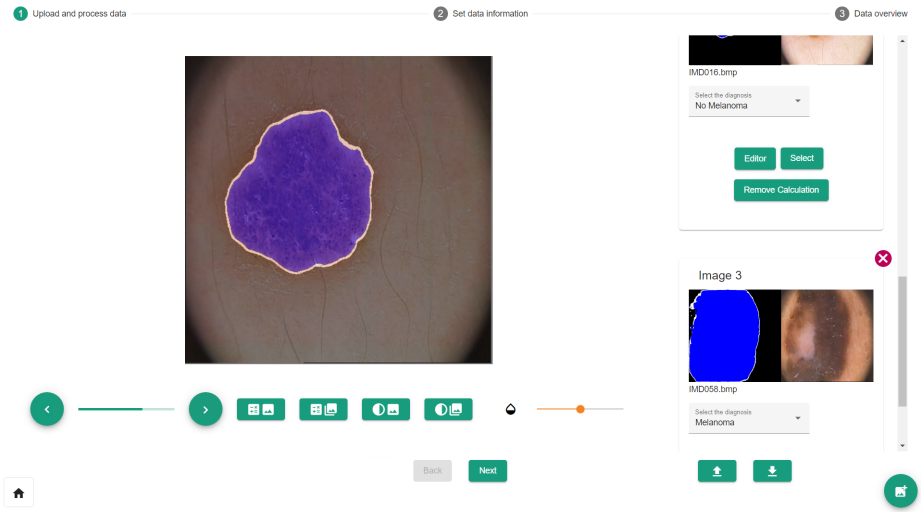


Fig. 4. Data uploader and editor

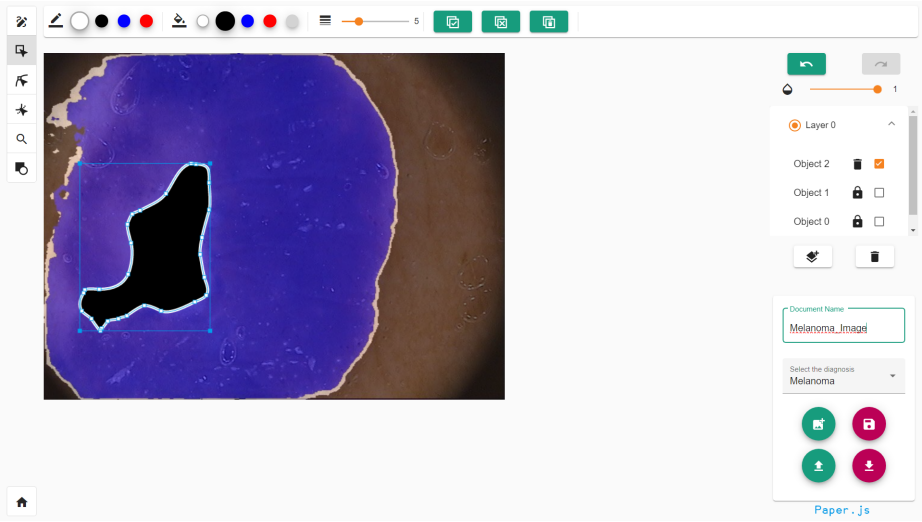


Fig. 5. Editor

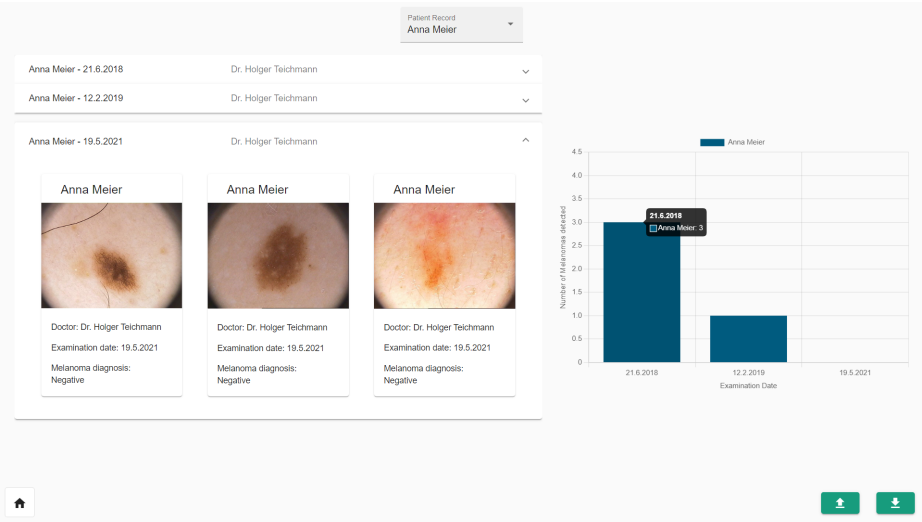


Fig. 6. Data visualization

is insufficient. In our case, the advantages of the web application clearly outweigh the disadvantages. Moreover, without a web application, the user would have to provide not only the editing software itself but possibly also the neural networks locally. This would require far too much computing effort. For the prototype, the range of functionalities that can be integrated into a web application is completely sufficient. Moreover, the connection to the Internet is constantly guaranteed from all workstations at our institute, and employees could thus access the application via the intranet without having to install it.

There are different ways to display image data, including raster images and vector graphics. The first method represents the image data as a two-dimensional array of data points, or pixels, with each pixel containing information about color, brightness, and location information. Compared to raster images, vector graphics are defined using objects like paths and shapes, where the information about those objects is described through mathematical formulas. Due to this type of representation, vector graphics can be scaled almost infinitely without compromising quality, like with raster images. Therefore, we analyzed different vector-based libraries for our application and decided to use Paper.js.

8 Conclusion and Future Work

In this paper, we researched the topic of how to implement an image prediction and evaluation based on the users' requirements, an extendable software structure, and usability criteria. We found out, that a web application provides us with lots of advantages compared to common desktop applications, especially in the field of research where applications like this can be applied due to restric-

tions in installation rights and hardware performance. Therefore, we developed a partially-automated web application that integrates image classification, segmentation, post-processing, and evaluation based on the melanoma use case. With the support of trained neural networks, the application can perform category prediction and automatic segmentation of lesion areas on images uploaded by the user, who can also participate in the image annotation process in post-processing or apply different evaluation methods to the data according to their needs. Inspired by the "human-in-the-loop" concept, the web application allows scientists to monitor and participate in the neural network decision-making process while ensuring the automated annotation process of images. This reduces the human error rate due to redundant manual labeling processes and iteratively improves the performance of the neural network by correcting and expanding the dataset. Due licensing restriction the source code cannot be shared publicly currently. In the future, we plan to expand the functionality of the web application in a modular way according to the needs of our users and the requested use case. We plan to evaluate the usability with other users for other use cases.

Acknowledgements

This work was supported by the Fraunhofer Internal Programs under Grant No. Attract 042-601000.

References

1. Arganda-Carreras, I., Kaynig, V., Rueden, C., Eliceiri, K.W., Schindelin, J., Cardona, A., Sebastian Seung, H.: Trainable weka segmentation: a machine learning tool for microscopy pixel classification. Bioinformatics **33**(15), 2424–2426 (2017)
2. Assael, Y.M., Shillingford, B., Whiteson, S., De Freitas, N.: Lipnet: End-to-end sentence-level lipreading. arXiv preprint arXiv:1611.01599 (2016)
3. Berg, S., Kutra, D., Kroeger, T., Straehle, C.N., Kausler, B.X., Haubold, C., Schiegg, M., Ales, J., Beier, T., Rudy, M., et al.: Ilastik: interactive machine learning for (bio) image analysis. Nature methods **16**(12), 1226–1232 (2019)
4. Budd, S., Robinson, E.C., Kainz, B.: A survey on active learning and human-in-the-loop deep learning for medical image analysis. Medical Image Analysis **71**, 102062 (2021)
5. Chaurasia, A., Culurciello, E.: Linknet: Exploiting encoder representations for efficient semantic segmentation. In: 2017 IEEE visual communications and image processing (VCIP). pp. 1–4. IEEE (2017)
6. Chen, L.C., Papandreou, G., Schroff, F., Adam, H.: Rethinking atrous convolution for semantic image segmentation. arXiv preprint arXiv:1706.05587 (2017)
7. Chen, L.C., Zhu, Y., Papandreou, G., Schroff, F., Adam, H.: Encoder-decoder with atrous separable convolution for semantic image segmentation. In: Proceedings of the European conference on computer vision (ECCV). pp. 801–818 (2018)
8. Chlebek, P.: Praxis der User Interface-Entwicklung: Informationsstrukturen, Designpatterns und Vorgehensmuster. Vieweg + Teubner Verlag, Wiesbaden, 1 edn. (2011)

9. 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. Ieee (2009)
10. Divito, S.J., Ferris, L.K.: Advances and short comings in the early diagnosis of melanoma. Melanoma research **20**(6), 450–458 (2010)
11. Fan, T., Wang, G., Li, Y., Wang, H.: Ma-net: A multi-scale attention network for liver and tumor segmentation. IEEE Access **8**, 179656–179665 (2020)
12. Gonzalez, R., Woods, R.: Digital Image Processing Global Edition. Pearson (2017)
13. Grechenig, T., Bernhart, M., Breiteneder, R., Kappel, K.: Softwaretechnik : mit Fallbeispielen aus realen Entwicklungsprojekten. Pearson Studium, München (2010)
14. Greenwald, N.F., Miller, G., Moen, E., Kong, A., Kagel, A., Dougherty, T., Fullaway, C.C., McIntosh, B.J., Leow, K.X., Schwartz, M.S., et al.: Whole-cell segmentation of tissue images with human-level performance using large-scale data annotation and deep learning. Nature biotechnology **40**(4), 555–565 (2022)
15. Guan, H., Liu, M.: Domain adaptation for medical image analysis: a survey. IEEE Transactions on Biomedical Engineering **69**(3), 1173–1185 (2021)
16. He, K., Zhang, X., Ren, S., Sun, J.: Deep residual learning for image recognition. In: Proceedings of the IEEE conference on computer vision and pattern recognition. pp. 770–778 (2016)
17. Iqbal, A., Sharif, M., Khan, M.A., Nisar, W., Alhaisoni, M.: Ff-unet: a u-shaped deep convolutional neural network for multimodal biomedical image segmentation. Cognitive Computation **14**(4), 1287–1302 (2022)
18. Isensee, F., Jaeger, P.F., Kohl, S.A., Petersen, J., Maier-Hein, K.H.: nnu-net: a self-configuring method for deep learning-based biomedical image segmentation. Nature methods **18**(2), 203–211 (2021)
19. Li, H., Xiong, P., An, J., Wang, L.: Pyramid attention network for semantic segmentation. arXiv preprint arXiv:1805.10180 (2018)
20. Lin, T.Y., Dollár, P., Girshick, R., He, K., Hariharan, B., Belongie, S.: Feature pyramid networks for object detection. In: Proceedings of the IEEE conference on computer vision and pattern recognition. pp. 2117–2125 (2017)
21. Lu, A.X., Zarin, T., Hsu, I.S., Moses, A.M.: Yeastspotter: accurate and parameter-free web segmentation for microscopy images of yeast cells. Bioinformatics **35**(21), 4525–4527 (2019)
22. Malcher, F., Hoppe, J., Koppenhagen, D.: Angular: Grundlagen, fortgeschrittene Themen und Best Practices. dpunkt.verlag, Heidelberg, 2 edn. (2019)
23. Marée, R., Rollus, L., Stévens, B., Hoyoux, R., Louppe, G., Vandaele, R., Begon, J.M., Kainz, P., Geurts, P., Wehenkel, L.: Collaborative analysis of multi-gigapixel imaging data using cytomine. Bioinformatics **32**(9), 1395–1401 (2016)
24. Meijering, E.: A bird's-eye view of deep learning in bioimage analysis. Computational and structural biotechnology journal **18**, 2312–2325 (2020)
25. Mendonça, T., Ferreira, P.M., Marques, J.S., Marcal, A.R., Rozeira, J.: Ph 2-a dermoscopic image database for research and benchmarking. In: 2013 35th annual international conference of the IEEE engineering in medicine and biology society (EMBC). pp. 5437–5440. IEEE (2013)
26. Olston, C., Fiedel, N., Gorovoy, K., Harmsen, J., Lao, L., Li, F., Rajashekhar, V., Ramesh, S., Soyke, J.: Tensorflow-serving: Flexible, high-performance ml serving. arXiv preprint arXiv:1712.06139 (2017)
27. Prümper, J.: Der benutzungsfragebogen isonorm 9241/10: Ergebnisse zur reliabilität und validität. Software-Ergonomie'97: Usability Engineering: Integration von Mensch-Computer-Interaktion und Software-Entwicklung pp. 253–262 (1997)

28. Richter, M., Flückiger, M.: Usability and UX kompakt: Produkte für Menschen. Springer Vieweg, Berlin, Heidelberg, 4 edn. (2016)
29. Ronneberger, O., Fischer, P., Brox, T.: U-net: Convolutional networks for biomedical image segmentation. In: Medical Image Computing and Computer-Assisted Intervention–MICCAI 2015: 18th International Conference, Munich, Germany, October 5-9, 2015, Proceedings, Part III 18. pp. 234–241. Springer (2015)
30. Sargent, D.J.: Comparison of artificial neural networks with other statistical approaches: results from medical data sets. Cancer: Interdisciplinary International Journal of the American Cancer Society **91**(S8), 1636–1642 (2001)
31. Schneider, C.A., Rasband, W.S., Eliceiri, K.W.: Nih image to imagej: 25 years of image analysis. Nature methods **9**(7), 671–675 (2012)
32. Seifert, C., Scherzinger, S., Wiese, L.: Towards generating consumer labels for machine learning models. In: 2019 IEEE First International Conference on Cognitive Machine Intelligence (CogMI). pp. 173–179. IEEE (2019)
33. Siegel, R.L., Miller, K.D., Fuchs, H.E., Jemal, A.: Cancer statistics, 2022. CA: a cancer journal for clinicians **72**(1), 7–33 (2022)
34. Sommerville, I.: Software Engineering. Pearson, München, 9 edn. (2012)
35. Stapelkamp, T.: Screen und Interfacedesign. Springer, Berlin, Heidelberg, 1 edn. (2007)
36. Steinmeyer, C., Dehmel, S., Theidel, D., Braun, A., Wiese, L.: Automating bronchoconstriction analysis based on u-net. In: EDBT/ICDT Workshops (2021)
37. Torralba, A., Russell, B.C., Yuen, J.: Labelme: Online image annotation and applications. Proceedings of the IEEE **98**(8), 1467–1484 (2010)
38. Werneburg, G.T., Werneburg, E.A., Goldman, H.B., Mullhaupt, A.P., Vasavada, S.P.: Neural networks outperform expert humans in predicting patient impressions of symptomatic improvement following overactive bladder treatment. International Urogynecology Journal pp. 1–8 (2022)
39. Wiese, L., Höltje, D.: Nncompare: a framework for dataset selection, data augmentation and comparison of different neural networks for medical image analysis. In: Proceedings of the Fifth Workshop on Data Management for End-To-End Machine Learning. pp. 1–7 (2021)
40. Zhao, H., Shi, J., Qi, X., Wang, X., Jia, J.: Pyramid scene parsing network. In: Proceedings of the IEEE conference on computer vision and pattern recognition. pp. 2881–2890 (2017)
41. Zhou, Z., Rahman Siddiquee, M.M., Tajbakhsh, N., Liang, J.: Unet++: A nested u-net architecture for medical image segmentation. In: Deep Learning in Medical Image Analysis and Multimodal Learning for Clinical Decision Support held in conjunction with MICCAI. pp. 3–11. Springer (2018)