NNCompare: A framework for dataset selection, data augmentation and comparison of different neural networks for medical image analysis

Lena Wiese*

lena.wiese@item.fraunhofer.de
Fraunhofer Institute for Toxicology and Experimental
Medicine
Hannover, Germany

Deborah Hölftje

deborah.hoeltje@item.fraunhofer.de
Fraunhofer Institute for Toxicology and Experimental
Medicine
Hannover, Germany

ABSTRACT

In our institute we capture a variety of medical image data – in particular, microscopy data for the use case of bronchoconstriction. In order to alleviate the manual intervention for the image analysis, we developed a comprehensive data analysis pipeline that automates the image preprocessing, data augmentation, as well as neural network training and deployment with a web-based user interface. We evaluate the pipeline on a real-world medical image dataset and comparatively analyze the performance of four different neural network architectures.

KEYWORDS

neural networks, medical images, image analysis

ACM Reference Format:

Lena Wiese and Deborah Hölftje. 2021. NNCompare: A framework for dataset selection, data augmentation and comparison of different neural networks for medical image analysis. In *International Workshop on Data Management for End-to-End Machine Learning (DEEM'21), June 20–25, 2021, Virtual Event, China*. ACM, New York, NY, USA, 7 pages. <https://doi.org/10.1145/3462462.3468884>

1 INTRODUCTION

Biomedical image analysis can be supported by neural networks specialized for image segmentation. Yet these networks have to be (re-)trained in order to be customized and optimized for each individual segmentation task. This training process usually requires provisioning a sufficiently large number of high-quality expert labels – which however often are not available in legacy systems. This fact is an obstacle to the application of neural networks in practice. Particularly in medical applications a high accuracy of the outcome segmentation is required, too, because often the subsequent decision (like evaluation of drug efficacy) has severe implications on patients. Thus any automated data analysis by neural networks

* Also with Institute of Computer Science, Goethe University Frankfurt.

Permission to make digital or hard copies of all or part of this work for personal or classroom use is granted without fee provided that copies are not made or distributed for profit or commercial advantage and that copies bear this notice and the full citation on the first page. Copyrights for components of this work owned by others than the author(s) must be honored. Abstracting with credit is permitted. To copy otherwise, or republish, to post on servers or to redistribute to lists, requires prior specific permission and/or a fee. Request permissions from permissions@acm.org.

DEEM'21, June 20–25, 2021, Virtual Event, China

© 2021 Copyright held by the owner/author(s). Publication rights licensed to ACM.
ACM ISBN 978-1-4503-8486-5/21/06...\$15.00
<https://doi.org/10.1145/3462462.3468884>

should be thoroughly tested for correctness, reproducibility and generalizability.

In the medical contexts at our institute different medical image data are obtained by modern imaging technologies like specialized high-resolution microscopes (for tissue analysis) or modern chip cytometers (for cell detection). To cater the needs of the different medical use cases, we need an end-to-end data analysis pipeline in which different neural network architectures can be flexibly exchanged and different run configurations can be specified and tested under unified comparable circumstances.

As our foremost use case we address bronchoconstriction analysis [18]. In this use case, so-called precision cut lung slices (PCLSs) are treated with medication in a petri dish. Images of the tissue slices are then captured at several distinct time steps. The significance of PCLSs is that the tissue is still vital and hence reacts to medication as if still being in a living organism. They are three-dimensional parts of tissue in which the bronchoconstriction analysis task is to identify airways in the tissue (and hence differentiate airways from blood vessels or vascular walls). With the identified airways the area of the airway can be computed to estimate the amount of air passing through it. The efficacy of bronchodilators (for example against asthma) can hence be assessed.

Apart from differences between the mentioned medical use cases (like tissue analysis versus cell detection), a further challenge is the variability of the image quality for a single use case. With our bronchoconstriction data set we face the following challenges. Within a single image there may exist:

- Light reflections (caused by the petri dish or some metal parts not covered by liquid medium)
- Shadows (caused by the petri dish or caused by the tissue itself depending on the angle of the airway with respect to the cut surface)
- Airways may lie (partially) in border regions of the PCLS
- Artifacts may be visually similar to airways (in particular blood vessels)

Moreover, because we observe the drug response over time, we obtain a series of images per PCLS. Yet, the observed petri dish is moved and readjusted during each treatment. Hence image quality is reduced by:

- Shifting the petri dish which alters the orientation of image
- Motion blur which may occur if an image is already taken while the petri dish is still being placed back or adjusted
- Changes of lighting conditions which may occur over time

- An airway being temporarily not measurable (e.g. out of focus)
- An airway being temporarily covered due to moving parts (like pipettes)

The current status quo of the image evaluation consists of multiple independently executed manual image processing steps. The currently used tools use the primitive factor of brightness for airway detection. This requires the manual adjustment of a threshold but still often provides suboptimal results. Hence we see in this use case as a lot of room for improvement in terms of accuracy, speed, objectivity and reliability of the analysis as well as the visualization of results as well as the usability of the whole solution.

Contributions. In this article we present our approach towards the development of an end-to-end image analysis framework that consists of a user-oriented web application where the input consists of raw image data that are analyzed in a highly automated pipeline and delivers back a resulting segmentation and analysis protocol to the web interface.

Outline. The paper is outlined as follows. Section 2 starts with an overview of the system architecture. Section 3 describes the internal data management and subset selection process as well as the data preprocessing. Section 4 introduces the configuration of neural network architectures as well as their training and evaluation. Section 5 presents the interfaces with which users can interact with our system. Section 6 comparatively analyzes four different network architectures. Section 7 discusses related work and Section 8 concludes the paper with a description of further extensions.

2 SYSTEM ARCHITECTURE

Based on the challenges raised in the introduction, several requirements have to be addressed by our system:

- **Robustness:** Being able to handle the high variety in image quality (color vs. grayscale, operators, resolution, contrast etc.)
- **Standardization:** Developing interfaces and (meta-)data formats to enable automation; in particular, making models accessible via a RESTful API to support the microservice paradigm
- **Modularization:** making ML model management independent from data management and making the training process independent from the productive deployment
- **Flexibility:** Deploying different ML systems on different hardware environments with only small configuration changes
- **Generality:** Supporting a variety of neural network architectures under varying configuration settings within a comparative approach that can easily be applied to different use cases at our institute.
- **Configurability/Customizability:** Being able to modify any component in the system and adapt it to the specific use case under investigation – even up to specific segmentation evaluation metrics
- **Reproducibility:** Monitoring data set specifics, training settings as well as a model export functionality
- **Optimization:** Supporting repeated executions by not reloading and rebuilding data sets for different runs on the same

data subset but with different network architectures or configuration settings.

Based on these requirements we decided to devise a software architecture that features a highly modular design as well as leverages customizability and reproducibility by storing configuration files for each of the modules. Figure 1 presents the high-level view of our framework comprising modules for storage and data management, subset creation, data preprocessing, neural network architectures and training process as well as segmentation evaluation; it also offers different user interfaces. The following sections provide more details of the modules of our framework. In addition, we devised an entire test suite with pytest that provides unit tests for the functionalities of each module.

3 DATA MANAGEMENT

For the bronchoconstriction use case we dispose of several PCLS images collected from experiments during several years and captured with different microscopes. Hence we have a high variability of conditions in the images (e.g. gray-scale images versus color images). Moreover, we are in the favorable situation to have the domain experts in the institute who manually labelled a subset of the data to provide us with a high-quality ground truth data set.

3.1 Data Labeling and Metadata

We chose a representative subset of the whole data collection for labeling: From each series of images (from one PCLS) we labeled the first one. Those images are representative because all cases are covered and the images later in the same series have little structural differences. Label data are stored as RGB images. Label colors for individual target classes can be specified in the metadata for each image. In the bronchoconstriction case, we have three main target classes each of which is associated with one tissue type in the images: blood vessel, airway or border of an airway; in addition we also consider the black image background as a target class.

We aim to identify the influence of the data quality challenges (as described in Section 1) on the prediction accuracy. While labeling the data set we hence also recorded metadata representing the data sample quality (e.g. regarding the skewness of the airway [18]). Those quality labels are not used in the training process itself but enable us to analyze different variants of the dataset (from datasets with only excellent quality up to datasets with both high and low quality images). The following criteria for image quality were considered:

- How good is the overall image contrast based on a qualitative comparison of the airway or blood vessel to the surrounding tissue?
- Was the microscope adjusted orthogonally to the plane in which the airway lies?
- Is the edge of the airway clearly visible (that is, not blurred)?
- Is the area of interest directly visible (that is, not partially covered up by tissue or particle contaminations nor distorted by light reflections).

Based on the expert decision regarding these criteria, we annotate each image with one of the following quality assessments: BAD: If zero or one criterion is met; GOOD: If two criteria are met; GREAT: If at least three criteria are met.

Figure 1: The system architecture of our framework

Hence internally we represent each data point by an array of the raw input image, the label containing colored pixels for each target class, as well as the JSON formatted metadata.

Overall we obtained a labeled dataset with the following statistics: A total of 420 labeled images of size 1038×1388 pixels containing four target classes (background, blood vessel, airway and airway border). As quality metadata 53 images were classified as great, 345 as good and 22 as bad.

3.2 Subset Creation

We need a flexible input data selection mechanism based on the quality criteria specified in Section 3. In this way we can filter out training data that are irrelevant or even counterproductive for the training process of the particular network. In our pipeline we automate the NN execution on different subsets of the training data as specified in metadata files.

On disk, we have separate folders for the raw data, the labeled data and the metadata. Specifications of the subset criteria are stored in a definition file. Hence, our subset creation module ensures that a specified dataset is ready for processing by reading in relevant data (data matching the definition criteria) from the folders into memory and creating the appropriate data point objects.

To assess created subsets from a statistical point of view, both the labeled as well as the raw data are analyzed:

- For the labeled data, the area (in pixels) per target class is computed on each image in the subset. More precisely, our framework calculates the relative area (represented by a number between 0 and 1) of the image that is covered by each target class based on its color. Over the whole subset the mean, min and max of the area is monitored. For example, the airway area ranges between 16 and 51 percent.
- For the raw data, the RGB image is converted into gray scale; then its brightness (mean gray value) as well as the contrast

(standard deviation of the gray values) are computed; moreover, the image is converted into HSV format and hue, saturation and value are computed.

This will allow us to assess the influence of statistical properties of each subset on the segmentation accuracy.

3.3 Data Preprocessing

Our framework includes a preprocessing module that can modify data “on the fly”. On the one hand, the module normalizes all input data by scaling the image intensities to the range between one and zero. In this way we reduce the influence of the variability of the data in terms of capture modes, lighting conditions and so on.

On the other hand, it offers resizing of the images by a given percentage while maintaining the original height to width ratio. With the help of this module we analyzed the influence of different scales on the segmentation accuracy [18]. Moreover, the resizing module can flexibly cater the different input shape requirements of the supported neural network architectures (see Section 4).

Furthermore, in order to increase the variety and amount of training data, we offer different augmentation options. When augmentation is enabled in the run configuration, the following augmentations are available:

- Rotating (maximum 45 degrees),
- shifting (maximum one tenth of image width or height),
- shearing (by at most 5 degrees to keep the general aspect of the airways intact),
- zooming (zooming in by at most 20 percent or zooming out by at most 10 percent) as well as
- flipping (horizontally or vertically).

All these augmentations are at the same time executed on the corresponding images. For filling the “holes” in the transformed image (originating from the augmentations) we fix “reflect” mode – it mirrors the nearby parts in the image to fill the holes. This is the

best option for the bronchoconstriction data, because it retains the natural structure of the airway tissue.

For each of these augmentations we generate random numbers to decide whether and how much the augmentation is applied. Hence, the resulting output image (and its corresponding label) consists of the union of all these augmentations. In addition, we also consider the two label disturbing augmentations of shifting the channel intensity and adjusting the brightness of the image. We only apply these to the raw image but not the label.

4 NEURAL NETWORK ARCHITECTURES

The aim of our framework is to comparatively analyze different neural network architectures on the same data sets. Hence, our neural network module allows us to “plug in” implementations of different neural network architectures. Each of the network architectures can be customized by neural network configuration files. Currently our framework incorporates some established encoder-decoder architectures: we offer a customizable U-Net [13] as well as widely used Tensorflow implementations available on Github¹ of FPN [6], Linknet [4] and PSPNet [20].

4.1 Network Configuration

Depending on the model architecture, different settings can be configured in configuration files. For the configurability of U-Net, configuration options include the amount of unique target classes in the segmentation output, which activation functions are used in internal layers as well as output layer, whether to use batch normalization across the channel axis between convolutional layers, whether to use average pool or max pool sampling in the encoder part, whether to use transposed convolutions or simple upsampling in the decoder part, the dimensions of pooling kernels, the amount of dropout after the initial convolutional block, the type of dropout (“spatial” or “standard”) to apply, whether to use dropout in the decoder part of the network, whether to use an attention dynamic [11] when concatenating with the skip-connection, the number of convolutional filters in the initial convolutional block as well as the number of total layers in the encoder.

The other architectures are somewhat less configurable due to the chosen library allowing for options for the specific encoder architecture, like the number of target classes and the type of output activation function.

Our framework can compile the models and store them together with metadata for the network configuration for later use during the training process.

4.2 Training Process

A training run can be customized to a high extent in our framework. A run configuration file specifies the compiled model to load, the parameter settings for optimizer and loss function as well as amount of epochs and batch size, early stopping and learning rate, and whether to use augmentation (see Section 3.3). In order to make use of the subset module, an identifier for the subset of the data to train on (see Section 3.2) has to be specified; a desired ratio for

test data as well as a desired ratio for validation data and a random seed for a shuffle operation on the data can be specified.

Further configuration options are available for cross validation (if desired): When specifying the parameter k , the training module splits the given dataset into k random subsets, each missing $1/k$ of the original set. This returns two parallel lists of corresponding batched train and test datasets; if the received dataset is not batched, it will be batched with a batch size of one.

After the training, the trained model is stored together with parameter settings and the time that the model took for training. The time of creation of the model is stored in a timestamp in order to support versioning and reproducibility. In addition, with our training module a pretrained model can be reloaded and training can be continued.

When training is finished, a so-called inference model can be exported together with information on the preprocessing steps applied; in particular, a resizing layer (see Section 3.3) is added that will adjust the input data to the shape expected by the trained model. With these metadata, the inference model can be applied to any (unseen) input image and a segmentation is returned. More precisely, our framework offers an inference-ready end-to-end method that takes a raw image (or a series of images) and returns properly formatted JSON response, suitable for REST. The response includes the model’s raw input (i.e., the preprocessed input images) and the model’s segmentation output.

4.3 Segmentation Evaluation

The evaluation module assesses the performance of the trained model on the test set. To evaluate the predicted (segmented) area compared to the label, a set of metrics can be specified; in particular, customized implementations of metrics can be added to our framework. Currently, mean intersection over union (where the average is computed over all target classes) and dice similarity coefficient (DSC) are computed. The segmentation evaluation module will compute all configured metrics for each segmentation. In addition, the mean and standard deviation of the metrics over all segmented files are monitored. The entire evaluation (including the raw data, the segmentation and the evaluation statistics) are stored together with information on the inference model used and with a timestamp.

5 USER INTERFACES

As part of our end-to-end system we developed appropriate user interfaces. For the end-users doing the biomedical experiments we devised an easy-to-use web-based interface. Whereas for the training setup the end-users are software developers and hence we devised a flexible command line interface (CLI) with a variety of available parameter settings.

More precisely, for the training configuration, a command line interface is available developed with the click² python package; in particular, a run configuration file and a model configuration file can be specified on the command line which are then parsed internally. If not only training but also testing should be performed the segmentation evaluation model is called by the CLI, too.

For the deployed neural network, we provide a flask app that for the case of a single image segmentation task lets users upload an

¹Pavel Yakubovskiy: Segmentation Models https://github.com/qubvel/segmentation_models

²<https://click.palletsprojects.com/>

Figure 2: Segmentation of image series and analysis of airway area as part of the Flask app

image for which then the segmentation is computed and displayed. Internally an inference model is pulled by Tensorflow Serving; as described in Section 4.2 the inference provides already a layer that takes care of all necessary preprocessing. The user can download and save the obtained segmentation as a jpeg. Moreover, the airway area is computed and displayed. For the task of segmenting an image series (see Figure 2), the user can upload a zip file of the image series; a plot of the changes in airway area is displayed next to the segmentation results. A further option is to store the segmentation output and airway area analysis on server side with a timestamp.

In our most recent developments, we devise an Angular-based web app with which end users can interactively improve the obtained segmentation: we provide a tool that stacks the layers of the raw image as well as the segmentations of the target classes on a drawing canvas on top of these layers allowing for a visual comparison of the raw image versus the segmentation mask. The features currently being added are: adding or removing annotations (for example for the image quality annotations), adjusting the segmentation boundaries based on fabric.js³ and using fill algorithms to complete the segmentation and switch target classes of wrongly colored segmentation parts.

With this interactive web app we have the option to let users correct inaccurate parts of segmentations as well as extend our existing labeled dataset through a human-in-the-loop approach. In this way we continuously obtain more high-quality labeled data from the experts using our tools. Moreover, the labeled data will at the same time be adjusted to domain shifts during real-world application in preclinical practice.

6 COMPARISON BENCHMARK

We analyzed the influence of the provided image augmentations (Section 3.3) and different network architectures (Section 4) on the metric values (mIoU and DSC) as well as the runtime measurements. All testruns are performed on an Intel(R) Xeon(R) Gold 6252 CPU and two NVIDIA Tesla T4 Graphics Processing Units. In order to ensure comparability we execute preprocessing on the training set (the original data and their corresponding labels) once – this includes normalizing them between 0 and 1 and random augmentation of each training image (that is, a rotation of up to 45 degree, shifting the pixels horizontally and vertically for a maximum of 10%, flipping the image horizontally or vertically, zooming in up to 20%

or out for up to 10% of the image or stretching the image on an axis with a shear angle of 5 degrees); we then store this training data set and use it for all network architectures as discussed in the following subsections. 10-fold cross validation is appropriately executed for the benchmarks. All corresponding results are summarized in Table 1.

6.1 Unet

In order to adapt the Unet to our use case, we used the Keras Unet library to implement a custom Unet according to Ronneberger et al. [13]. The network is trained with 4 Unet layers, image resolution of 512×512 , batch size of 4, ReLU as activation function, 64 filters, transposed upsampling convolutions and max-pooling for downsampling. Training this network results in a mIoU of 93.14% with and 93.17% without data augmentation (for Dice this is 96.34% versus 96.36%, respectively). The training time, however increases from 21 minutes without augmentation to 95 minute when applying it. Considering the insignificant improvement of 0.03% mIoU, training without augmentation is preferred here.

The Unet has difficulties recognizing those areas of the airway which are not completely black. This happens whenever the airway is not straight but skewed and the microscopy image therefore also shows the wall of the airway. Additionally, it tends to recognize further black areas (like background or blood vessels) in the image as airways as well (False Positives).

6.2 PSPNet

Next, we evaluate the Pyramid Scene Parsing Network (PSPNet) [20] with the aim to improve the segmentation results on our bronchoconstriction dataset. Preprocessing of the data consists of resizing the images to a resolution of 720×720 pixels.

For the PSPNet architecture we use the segmentation models package (see Section 4), which provides a Tensorflow implementation. To account for the depth of the network, we applied a ResNet101 encoder (“backbone”) for the PSPNet. Besides, we chose a batch size of 8, learning rate of 0.0001 and early stopping in the case that 12 epochs in a row show no improvements. Interestingly, PSPNet showed a slightly worse performance in terms of mIoU and Dice Similarity Coefficient in comparison to the customized Unet; but a significantly reduced runtime during training. Again, augmentation resulted in only a slight improvement.

³<http://fabricjs.com/>

Net	With Augmentation			Without Augmentation		
	mIoU	dice	training	mIoU	dice	training
Unet	93.14	96.34	95 min	93.17	96.36	21 min
PSPNet	92.95	96.27	19 min	92.62	96.07	9 min
LinkNet	93.43	96.52	64 min	93.30	96.44	30 min
FPN	94.16	96.91	76 min	93.98	96.82	55 min

Table 1: Mean IoU and DSC for all networks with and without data augmentation

Looking at the prediction masks, it is noticeable that the PSPNet also shows difficulties recognizing skewed airways not lying perfectly orthogonal to the focus plane. Additionally, the PSPNet does not show any False Positives on the segmentation masks but shows more False Negatives than the Unet.

6.3 LinkNet

Dataset preprocessing follows the same steps, except for the resolution, which is 800×800 . As a Tensorflow implementation of LinkNet [4] we also used the segmentation models package (see Section 4). Furthermore, a ResNet101 encoder backbone is included here, too. The weights are initialized with on imageNet pretrained weights and we chose a batch size of 4, learning rate of 0.001 and early stopping parameter of 12. Additionally, batch normalization is applied in the decoder block. The same loss as in Section 6.2 is applied here. The training time without using augmentation is 30 minutes, while with augmentation we get 64 minutes, which are both substantially higher than for the PSPNet. Both networks achieve a similar mIoU of 93.30 % without augmentation and 93.43 % with augmentation (and respectively for Dice 96.44% versus 96.52%) again confirming that the augmentation is insignificant here.

Prediction masks of the LinkNet do not recognize the boundary of the airways (recall that airway boundary is a target class in our networks). Otherwise, it has less difficulties with skewed airways than the PSPNet, however it is still not recognizing skewed airways in some images. Further, the LinkNet also shows more false positives (some black areas are incorrectly also classified as airway) than false negatives.

6.4 FPN

The Feature Pyramid Network (FPN) [6] follows the same preprocessing and we also use a ResNet101 encoder backbone, batch normalization, imageNet pre-trained weights, batch size of 4, learning rate of 0.001 and early stopping parameter of 12. Further, we use 256 filters in the feature pyramid blocks and aggregate them using concatenation. While having a training time of 55 minutes without augmentation and 76 minutes with augmentation, which is roughly twice as much as the LinkNet training time, we get insignificant mIoU improvements of 93.98 % without augmentation and 94.16 % with augmentation (and respectively for Dice 96.82% versus 96.91%).

The prediction masks of FPN show no airway boundaries (as LinkNet), has the same difficulties classifying skewed airways and shows false positives.

7 RELATED WORK

Finding optimal settings for (hyper-)parameters in machine learning environments is an inherent and complex problem in real-world environments. Predictive performance is highly dependent on data inputs. In this sense, [15] state that “[t]raditional abstractions and boundaries may be subtly corrupted or invalidated by the fact that data influences ML system behavior.” With our system we aim to follow their principles of a “good configuration system” – in particular regarding the ease “to specify a configuration as a small change from a previous configuration” [15] due to our modular approach and the separation of run configuration and model configuration. Our comparative approach also helps to at least observe and monitor the CACE principle: “Changing Anything Changes Everything” [15] because we track configurations as well as outcomes continuously.

Related work like [9, 10] focuses on parallel model selection but less on the end-to-end pipeline including data selection and data augmentations. Moreover, other approaches are not valid for image analysis; for example, [17] focus on data extraction for textual data and appropriate labeling. In the same vein, the TFX system [3] was not easily adaptable to our image analysis use case.

We also looked at AutoML systems as surveyed in [19] with the result that none of them offered helpful support for our medical image analysis task. As opposed to using AutoML as a blackbox system, we also had to always be able to fully investigate the internals of our system and discuss all intermediate steps with our medical colleagues in order to closely monitor the correct processing of our pipeline.

In medical applications, explainability [2] and transparency of the automated decisions are of major importance. Hence any bias [7, 14] present in the training data should be detected and it must be avoided that such a bias influences machine decisions. With the statistics, configurations and outcomes collected in our system we could hence go one step further and turn the collected data into consumer labels [16], model cards [8] or data sheets [5] for the trained machine learning models.

8 DISCUSSION AND CONCLUSION

In this paper we described our current platform that allows for comparison of different neural network architectures executed on medical image data. Major benefits are on the one hand the easy interaction for end users as well as a flexible configurability for the developers. We opted for a customized implementation (instead of using existing but less flexible tools) to give us the option of a modular approach and the possibility to include a wide range of advanced features. In current work we aim at the following added features:

- Extending our analysis to other medical image data sets like the mentioned cytometry data. For these datasets appropriate expert labels still have to be obtained.
- Using other image specific metrics for segmentation evaluation or even as alternative loss functions during the training process. Currently we are integrating the following three metrics from the Deep Mind Surface Distances package⁴ into our framework: the Average Surface Distance, the robust

⁴<https://github.com/deepmind/surface-distance>

Hausdorff distance, as well as Surface Dice. In case we observe no improvement with the mentioned metrics, we will also look at more recent advances like balanced Hausdorff distance [1].

- Carefully investigating the reason why the above mentioned augmentation fails to improve results as well as developing customized augmentation methods that help us get more training data for the difficult instances of the target classes. Based on the observation that the conventional augmentations do not achieve better segmentations, we aim to specifically look at the existing false positive and false negative cases by automatically identifying these difficult cases with the above mentioned image specific metrics. More precisely, we will be able to identify ‘problematic’ images inside the dataset based on comparing the metric values between classes and different configurations. Correlations between metric values may help to extend our approach from a manual analysis to an automatic quality control for a greater number of images. These problematic cases will then be used as a basis for customized data augmentations.
- Adding more support for MongoDB. We aim to store all JSON formatted configuration and metadata files in a MongoDB instance running on one of our CPU servers to improve the management of our configuration files and, with the application of database indexes, improve the version management of the trained models as well as segmentation evaluations.
- Investigating the influence of statistical properties of the subsets on the segmentation output. Based on the statistical analysis described in Section 3.2 we want to detect domain shifts during the real-world deployment of our framework and hence allow for a re-training of the neural networks whenever enough new labeled data have been supplied by end users.
- Testing more and different configurations of the included network architectures or devising a more customized network architecture for our specific use case. One approach to improve segmentations would be to follow a specialized coarse-to-fine approach [12] for blurry image regions or to address boundary refinement [21]. In particular, we also want to be able to include state-of-the-art pytorch implementations – in particular, due to the fact that more segmentation models are available in a pytorch version⁵; as well as more models from the Tensorflow model garden⁶.

In summary, the close interaction with end users at our institute give us the opportunity to obtain expert labels and the development of our NNCompare tool lets us flexibly react to new requirements. This close cooperation will ensure a sustainable deployment of our platform.

ACKNOWLEDGMENTS

This work was supported by Fraunhofer Internal Programs under Grant No. Attract 042-601000. We thank C. Steinmeyer, D. Theidel, S. Dehmel and E. Spies for their support in the software implementation, documentation and data preparation.

⁵Pavel Yakubovskiy: https://github.com/qubvel/segmentation_models.pytorch

⁶<https://github.com/tensorflow/models>

REFERENCES

- [1] Orhun Utku Aydin, Abdel Aziz Taha, Adam Hilbert, Ahmed A Khalil, Ivana Galinovic, Jochen B Fiebach, Dietmar Frey, and Vince Istvan Madai. 2021. On the usage of average Hausdorff distance for segmentation performance assessment: hidden error when used for ranking. *European Radiology Experimental* 5, 1 (2021), 1–7.
- [2] Alejandro Barredo Arrieta, Natalia Díaz-Rodríguez, Javier Del Ser, Adrien Ben- netot, Siham Tabik, Alberto Barbado, Salvador García, Sergio Gil-López, Daniel Molina, Richard Benjamins, et al. 2020. Explainable Artificial Intelligence (XAI): Concepts, taxonomies, opportunities and challenges toward responsible AI. *Information Fusion* 58 (2020), 82–115.
- [3] Denis Baylor, Eric Breck, Heng-Tze Cheng, Noah Fiedel, Chuan Yu Foo, Zakaria Haque, Salem Haykal, Mustafa Ispir, Vihan Jain, Levent Koc, et al. 2017. TFx: A tensorflow-based production-scale machine learning platform. In *Proceedings of the 23rd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*. 1387–1395.
- [4] Abhishek Chaurasia and Eugenio Culurciello. 2017. Linknet: Exploiting encoder representations for efficient semantic segmentation. In *2017 IEEE Visual Communications and Image Processing (VCIP)*. IEEE, 1–4.
- [5] Timnit Gebru, Jamie Morgenstern, Briana Vecchione, Jennifer Wortman Vaughan, Hanna Wallach, Hal Daumé III, and Kate Crawford. 2018. Datasheets for datasets. *arXiv preprint arXiv:1803.09010* (2018).
- [6] Tsung-Yi Lin, Piotr Dollár, Ross Girshick, Kaiming He, Bharath Hariharan, and Serge Belongie. 2017. Feature pyramid networks for object detection. In *Proceedings of the IEEE conference on computer vision and pattern recognition*. 2117–2125.
- [7] Ninareh Mehrabi, Fred Morstatter, Nripsuta Saxena, Kristina Lerman, and Aram Galstyan. 2019. A survey on bias and fairness in machine learning. *arXiv preprint arXiv:1908.09635* (2019).
- [8] Margaret Mitchell, Simone Wu, Andrew Zaldivar, Parker Barnes, Lucy Vasserman, Ben Hutchinson, Elena Spitzer, Inioluwa Deborah Raji, and Timnit Gebru. 2019. Model cards for model reporting. In *Proceedings of the conference on fairness, accountability, and transparency*. 220–229.
- [9] Supun Nakandala, Yuhao Zhang, and Arun Kumar. 2019. Cerebro: Efficient and reproducible model selection on deep learning systems. In *Proceedings of the 3rd International Workshop on Data Management for End-to-End Machine Learning*. 1–4.
- [10] Supun Nakandala, Yuhao Zhang, and Arun Kumar. 2020. Cerebro: A data system for optimized deep learning model selection. *Proceedings of the VLDB Endowment* 13, 12 (2020), 2159–2173.
- [11] Ozan Oktay, Jo Schlemper, Loic Le Folgoc, Matthew Lee, Mattias Heinrich, Kazunari Misawa, Kensaku Mori, Steven McDonagh, Nils Y Hammerla, Bernhard Kainz, et al. 2018. Attention u-net: Learning where to look for the pancreas. *arXiv preprint arXiv:1804.03999* (2018).
- [12] Jose Jaena Mari Ople, Pin-Yi Yeh, Shih-Wei Sun, I-Te Tsai, and Kai-Lung Hua. 2020. Multi-scale neural network with dilated convolutions for image deblurring. *IEEE Access* 8 (2020), 53942–53952.
- [13] Olaf Ronneberger, Philipp Fischer, and Thomas Brox. 2015. U-net: Convolutional networks for biomedical image segmentation. In *Lecture Notes in Computer Science (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics)*, Vol. 9351. Springer Verlag, 234–241.
- [14] Stefanie Scherzinger, Christin Seifert, and Lena Wiese. 2019. The Best of both Worlds: Challenges in Linking Provenance and Explainability in Distributed Machine Learning. In *2019 IEEE 39th International Conference on Distributed Computing Systems (ICDCS)*. IEEE, 1620–1629.
- [15] David Sculley, Gary Holt, Daniel Golovin, Eugene Davydov, Todd Phillips, Dietmar Ebner, Vinay Chaudhary, Michael Young, Jean-Francois Crespo, and Dan Dennison. 2015. Hidden technical debt in machine learning systems. *Advances in neural information processing systems* 28 (2015), 2503–2511.
- [16] Christin Seifert, Stefanie Scherzinger, and Lena Wiese. 2019. Towards Generating Consumer Labels for Machine Learning Models. In *2019 IEEE First International Conference on Cognitive Machine Intelligence (CogMI)*. IEEE, 173–179.
- [17] Vraj Shah and Arun Kumar. 2019. The ML data prep zoo: Towards semi-automatic data preparation for ML. In *Proceedings of the 3rd International Workshop on Data Management for End-to-End Machine Learning*. 1–4.
- [18] Christian Steinmeyer, Susann Dehmel, David Theidel, Armin Braun, and Lena Wiese. 2021. Automating Bronchoconstriction Analysis based on U-Net. In *DARLI-AP@EDBT*.
- [19] Iordanis Xanthopoulos, Ioannis Tsamardinos, Vassilis Christophides, Eric Simon, and Alejandro Salinger. 2020. Putting the Human Back in the AutoML Loop. In *ETLMP@EDBT/ICDT Workshops*.
- [20] Hengshuang Zhao, Jianping Shi, Xiaojuan Qi, Xiaogang Wang, and Jiaya Jia. 2017. Pyramid scene parsing network. In *Proceedings of the IEEE conference on computer vision and pattern recognition*. 2881–2890.
- [21] Peng Zhou, Brian Price, Scott Cohen, Gregg Wilensky, and Larry S Davis. 2020. Deepstrip: High-Resolution Boundary Refinement. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*. 10558–10567.