Automating Bronchoconstriction Analysis based on U-Net

[Industrial and Application paper]

Christian Steinmeyer
Susann Dehmel
David Theidel
Armin Braun
Lena Wiese*
christian.steinmeyer@item.fraunhofer.de
susann.dehmel@item.fraunhofer.de
david.theidel@item.fraunhofer.de
armin.braun@item.fraunhofer.de
lena.wiese@item.fraunhofer.de
Fraunhofer Institute for Toxicology and Experimental Medicine
Hannover, Germany

ABSTRACT

Advances in deep learning enable the automation of a multitude of image analysis tasks. Yet, many solutions still rely on less automated, less advanced processes. To transition from an existing solution to a deep learning based one, an appropriate dataset needs to be created, preprocessed, as well as a model needs to be developed, and trained on these data. We successfully employ this process for bronchoconstriction analysis in Precision Cut Lung Slices for pre-clinical drug research. Our automated approach uses a variant of *U-net* for the core task of airway segmentation and reaches (mean) Intersection over Union of 0.9. It performs comparably to the semi-manual previous approach, but is approximately 80 times faster.

KEYWORDS

Image segmentation, Preprocessing pipeline, Medical image analysis, Bronchoconstriction, Lung tissue slices

1 INTRODUCTION

Inflammatory and allergic lung diseases, such as lung injury, pneumonia, asthma, chronic obstructive pulmonary disease, or pulmonary hypertension are still difficult to treat. A common symptom of these diseases is the obstruction of the airways or Airway Hyperresponsiveness (AHR). Bronchodilators aim to avoid or reduce these symptoms.

With advances in the field of deep learning, computers reach and sometimes surpass human level performance in specific tasks like image analysis [7, 25]. In semantic segmentation, Convolutional Neural Networks (CNNs) achieve state of the art results [12]. In this work, we describe our process of changing from a conventional, semi-manual workflow to a fully automated one using Neural Networks (NNs) for the evaluation of bronchodilators in microscopy image data from Precision Cut Lung Slices (PCLSs). We focus on the underlying dataset curation, image labelling, and processing. We present an optimized data processing pipeline from raw images to model input. We transparently develop a NN model for the task of semantic segmentation of airways. We also explore image quality as means to improve predictions. Finally we show preliminary results.

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

2 BACKGROUND

There is a continuous need of innovative or advanced drugs to treat the symptoms of obstructive lung diseases. PCLS is one of the developed models to evaluate bronchodilatory compounds in the non clinical phase of drug development. Bronchoconstriction analysis with this model is applied in house since 2012. PCLS allows to target small airways of the lung in different animals, (mouse, rat, or non-human primates) or human lung explant tissue. They can be analyzed to assess potency of bronchodilatory drugs through concentration-response curves. Mean inhibitory concentrations (IC_{50} values) can be compared to reference drugs within one human donor or animal to compare efficacy of the drugs [13, 18, 22, 24]. Bronchoconstriction can also be compared across different species (e. g., for AHR in asthma) [9]. In 13 studies, we tested bronchodilators or bronchoconstricting compounds and mode of action of drugs as contract research with pharmaceutical industry or within public studies (sponsored by BMBF, BMWI, DFG).

2.1 Image Acquisition

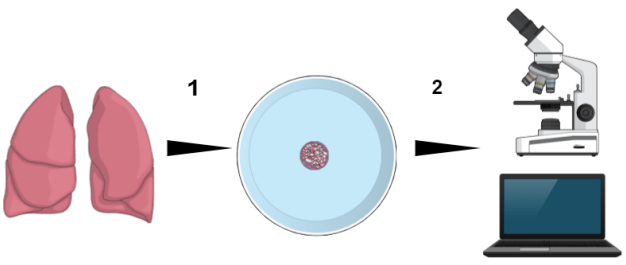


Figure 1: Experiment setup for image acquisition.

We use images that were acquired in previous studies testing efficacy of bronchodilatory drugs. In those studies, PCLSs were prepared from tumor-free parts of lung explants or from freshly isolated lungs after necropsy of animals, as described in [16] (see Figure 1, step 1). Lung lobes were cut into approximately 300 µm thick slices and transferred into petri dishes. Airways within PCLSs were imaged and digitized using an inverted microscope and a digital video camera (see Figure 1, step 2). Camera control and image analysis were achieved by *AxioVision* software (Zeiss, Germany). Each experiment setup differs in compound(s), doses, or specimen and is repeated with at least two samples. Throughout an experiment, images are taken in regular intervals. The resulting data are image series of varying length depending on the experiment (e. g., reaction times). In approximately 1750 such series from around 420 different experiment setups, over 55000 images are available.

2.2 Semi-Manual Airway Segmentation

In our previous studies, we used a semi-manual approach to analyze the microscopy images. The *Image J* plugin "*PCLS Area Tool*" was used for the airway segmentation. It is based on the *IsoData* approach [19], that uses different brightness levels between object of interest and background[1]. It has an automatic option (that we call "*IsoData*", here), as well as one with manual intervention (that we call "*IsoData+*"). The automatic option does not yield satisfactory results (see Section 6), mainly because it does not include locality. If surrounding tissue has similar luminance as the airway, a frequent result is a correctly segmented airway, surrounded by this incorrectly segmented other tissue. It can be vastly improved by manually setting the threshold and defining a region of interest around the airway. But as this takes time, we set out to automate this segmentation task.

3 RELATED WORK

In recent years, NNs have caused big leaps forward in semantic segmentation tasks [21, 26]. In 2015, Ronneberger et al. published the *U-net* architecture for image segmentation in a biomedical setting [20]. It was designed to cope with few training data and high resolutions, typical for medical image analysis. It consists of an encoder and a decoder. The encoder serves as a feature detector. Through a series of convolutions and max pool operations, it reduces the resolution (x and y dimensions) of the data while increasing its depth (number of feature maps). The decoder serves as an upsampler. Through a series of convolutions and deconvolutions, it increases the resolution again while decreasing the depth. It also contains skip connections from each layer before max pooling (encoder) to the corresponding layer after deconvolution (decoder). Finally, in the last layer, the output is mapped to the desired number of classes.

In the context of lungs, *U-net* is frequently used. It can be extended to fit specific tasks, e. g., through adding residual and recurrent connectivity [2], combining it with other architectures [4, 11], or extending it with a third dimension [11, 15]. We therefore select it as the starting point for our model. For a more generic review of deep learning techniques for semantic segmentation, see [10].

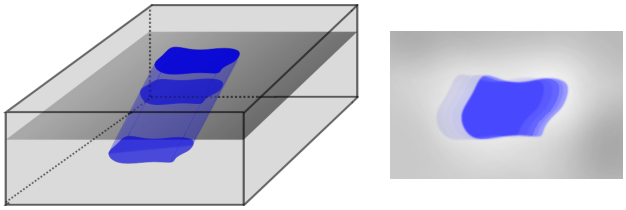
[1]For details on *IsoData*, see https://imagej.net/Auto_Threshold.html#IsoData (last accessed on 2021–02–14).

4 AUTOMATING SEGMENTATION

To automate airway segmentation with NNs, we need labelled data for supervised training. To the best of our knowledge, no public dataset for airway segmentation exists. Thus, we create our own (Section 4.1) and propose a data pipeline for preprocessing (Section 4.2). In Section 4.3, we describe the development of an ML model for semantic segmentation and analyse it in Sections 4.4 and 4.5.

4.1 Image Labelling

Because we are working towards the goal of optimal automatic segmentation, the quality of data labels is essential. The upper limit of model performance depends on the label quality. Deviation from the ground truth means a loss in model potential. Because the existing tools yield unsatisfactory labels, the image labelling is performed manually: Each label is created by a trained specialist. It is then reviewed and possibly adjusted by another one.



(a) 3D representation. The dark plane represents the camera's focus plane.

(b) 2D representation from above, similar to the camera perspective.

Figure 2: Schematic depiction of a PCLS section (grey), containing an airway (blue).

The main target of bronchoconstriction analysis is the relative change of airway volume. Our annotations resemble an approximation of the ground truth: We consider airways simplified as prisms (see Figure 2a). The volume V of a prism is defined as $V = b * h$, where b is the base area and h is the height of the prism. The airway cross-section surface is proportional to the airway volume; thus, we can use it as an indicator for the volume. If an airway lies perfectly orthogonal to the camera, all cross-section surfaces overlap. If not, cross-sections of different depths are shifted on the image plane (see Figure 2b). Some cross-sections might be out of focus and choosing an arbitrary cross-section might lead to a less stable model. To overcome these issues, we label the cross-section in the focus plane, i. e., the plane that lies parallel to the camera and is the sharpest. This includes more detail and allows more unambiguous labels. We create labels as *RGB* images representing class membership on pixel level, where the target class is one of *background*, *border*, or *airway*. Each target class is assigned a unique *RGB* color with maximal distance to other classes.

The labelled dataset should resemble real life data. In order to select a diverse dataset from as few images as possible, we choose one image per experiment setup as described in Section 2.1, yielding 420 samples.

4.2 Data Pipeline

The unprocessed sample data is preprocessed as depicted in the pipeline in Figure 3 before being fed to a model. First,

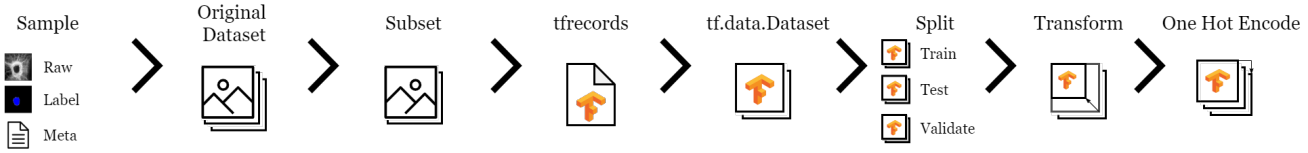


Figure 3: Data preprocessing pipeline from a sample to model input tensor.

one sample is defined as a raw image, its corresponding label created as described in Section 4.1, as well as its meta data (e. g., quality, see Section 5). The samples are collected in what we call the original dataset. From this original set, we can create a subset (e. g., by quality). This step is optional and depends on the goals of the test run.

Since working with image data is resource intensive, we aim to optimize computation time and cost. To that end, we store the dataset variants in the *tfrecords* format[2]. We read those files into *tensorflow.data.Dataset*[3] data structures for further processing. Both are part of the *tensorflow* framework [1] and highly optimized [8]. They support lazy evaluation, i. e., letting us read data on demand, reducing memory needs and allowing us to work with bigger images. We also use them to minimize idle times of Graphics Processing Units (GPUs). We transform the data to tensors. We do it early, because that way, we can profit most from *tensorflow*'s computation graph optimizations [8]. In most test runs, the data is then shuffled and split into train, test, and validation sets. Our model expects input tensors of a fixed shape. Tensor shape is defined by the image resolution (width, height) and number of (color) channels (depth). Some of the available images are colored. As the color channels might store useful information, we decode all input images to three channel *RGB* format. While in other domains, image tiling is applied to reduce model size [20], in our case the area covered by airways varies and often takes up the majority of an image. Thus, we do not apply tiling, but bilinearly transform them to have fixed dimensions (exact dimensions vary by experiment).

We want to use class based metrics (see Section 4.4) and hence one hot encode the label images. To that end, we transform them from *RGB* space to *HSV* space. We apply the same transformation to the target class representations described in Section 4.1. We use thresholds for hue (10), saturation (100) and value (100) to determine whether or not a pixel belongs to a target class. Each pixel is assigned at most to one target class that way. For each label, we use this to create binary masks for the classes. Finally, we concatenate the binary masks as channels, forming the one hot encoded labels (see last step in Figure 3).

4.3 Model Development Process

Within the scope of the model development process, we work towards fast exploration and experimentation. Maintaining high code quality and avoiding bugs are further meta goals. Hence, we employ the following mechanisms. Our model development process can be described as two intertwined cycles of research on one hand and development on the other (see Figure 4). We start by phrasing a question like "*How does image size influence model performance?*". If that question can be answered with the current code basis, we continue with the research cycle. We define the necessary test run parameters and configuration, before executing it in our research environment. This also serves the purpose of documentation and increases reproducibility as the same configuration can be run again. We use jupyter notebooks [14] to enable interaction with the results. As soon as new questions arise, we repeat the cycle.

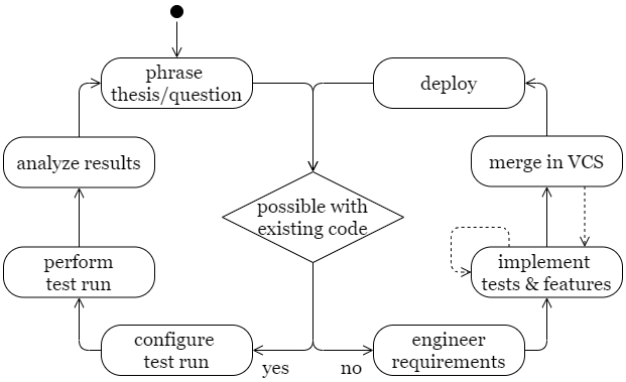


Figure 4: Research and development cycle. VCS stands for Version Control System.

If a question cannot be answered with the current code basis, we continue with the software development cycle. Driven by reusability and modularization, we formulate generalized requirements from the question (e. g., resize images to a given size). We then implement and test the missing features using test driven development [5]. In an effort to fail cheap and fast, we create unit and integration tests for our custom code with *pytest*[4]. In combination with syntactic checks, they ensure intended behavior. We integrate them in a Continuous Integration (CI) pipeline [17]. As a result, breaking changes are noticed and errors identified quickly (see Figure 4). All code is managed in a Version Control System (VCS). Once the features work locally, they are merged in the remote repository of the VCS. This triggers our CI pipeline, ensuring proper behavior in the test environment. If any of the tests fail, we repeat the implementation and following steps (cf. dotted arrows in Figure 4). On success, we deploy the new code base and continue with the research cycle.

[2]For details on *tfrecords*, see https://www.tensorflow.org/tutorials/load_data/tfrecord#setup (last accessed on 2021–02–14).
[3]For details on *tf.data.Dataset*, see https://www.tensorflow.org/api_docs/python/tf/data/Dataset (last accessed on 2021–02–14).
[4]For details on *pytest*, see https://docs.pytest.org/en/stable/ (last accessed on 2021–02–14).

4.4 Metrics and Defaults

In order to compare performance of different models or settings, we use numeric metrics. In our semantic segmentation task, we use per-pixel labelling. Due to strong class imbalance (usually there is more background than foreground), simply using accuracy does not suffice. By choosing metrics that consider both, the number of correct predictions, as well as the ratio of classes, they better represent the actual prediction quality. Due to its wide spread use [10] and its intuitive definition, we use (mean) Intersection over Union (mIoU), or Jaccard Distance, defined as:

$$mIoU = \frac{1}{n+1} \sum_{i=0}^{n} \frac{TP_i}{TP_i + FP_i + FN_i} \tag{1}$$

where n is the number of target classes, TP_i is the number of true positive, FP_i the number of false positive, and FN_i the number of false negative pixels for the target class i.

We also use (mean) Dice Similarity Coefficient (mDSC), also called quotient of similarity, *F1-score*, or harmonic average between precision and recall (cf. [2]), defined as:

$$mDSC = \frac{1}{n+1} \sum_{i=0}^{n} \frac{2TP_i}{2TP_i + FP_i + FN_i} \tag{2}$$

We use Categorical Cross Entropy (CCE) as both, loss and metric (lower is better), defined as:

$$CCE = -\sum_{j=0}^{m} y_j * \log p_j \tag{3}$$

where m is the total number of values (pixels through all channels), p_j is the model prediction for index j, and y_j is the corresponding target value.

In the upcoming experiments, we consider different variables, like resolution, and their effect on prediction performance. We use the following default setting, unless specified otherwise: (1) U-net with two encoder and decoder blocks (2) image dimensions of 128 by 128 (3) 64 filters in first convolutional block (4) spatial dropout of 0.5 in encoder (5) ReLU as activation function in inner layers (6) SoftMax as activation function in output layer (7) CCE as loss function (8) batch normalization between convolutional layers (9) training for 50 epochs with a batch size of 8.

Further, we use a variant of ten-fold cross validation: First, we split the data into train, test, and validation sets. For training, we then further split the train set into ten separate folds, each missing a different tenth of the original set. We use the validation set to observe training performance. After training, we use the test set to obtain results (i. e., the ten different folds are evaluated on the same test data). The reported metrics refer to either the average (CCE, mDSC, mIoU) over all folds, or sum (duration). All test runs are performed on an *Intel(R) Xeon(R) Gold 6252 CPU* and two *NVIDIA Tesla T4 GPUs*.

4.5 Experiments and Results

The approach described in Sections 4.2 and 4.3 enables us to efficiently perform experiments. We can evaluate hypotheses in fast cycles and robustly implement missing functionality. We start with small images. Then, we identify and address issues that arise as we scale up. In a first experiment, the input images and labels are bilinearly resized to differently sized squares (for images of size 1024, batch size is reduced to 4). The results are depicted in Table 1. The best results are achieved when using the smallest images, increasing image size leading to a decreasing performance.

Image Size	CCE	mDSC	mIoU	Duration
32	0.124	**0.951**	**0.908**	**0:14:55**
64	**0.118**	0.948	0.904	0:15:40
128	0.128	0.942	0.893	0:26:16
256	0.168	0.925	0.866	0:54:34
512	0.255	0.877	0.786	3:18:38
1024	0.313	0.852	0.749	13:56:10

Table 1: Experiment results for different image sizes, trained for 50 epochs each (Duration in hours:minutes:seconds)

We conclude that the task of airway segmentation is easier, if the image resolution is smaller, i. e., there are fewer details available. Also, models with smaller image sizes converge faster, which we show in another experiment: We increase the maximum number of epochs, but stop training early, if the model converges. To evaluate convergence, we consider validation loss and stop learning when there is no improvement for twelve epochs. The results are depicted in Table 2. When comparing the average duration in both experiments, we see: While models trained on images smaller than 128 pixels stop training earlier than 50 epochs, those trained on bigger images need more epochs before stopping. This also improves the results for bigger image size.

Image Size	CCE	mDSC	mIoU	Duration
32	0.113	0.947	0.902	**0:10:00**
64	**0.110**	**0.948**	**0.903**	0:13:54
128	0.119	0.945	0.898	0:26:18
256	0.148	0.936	0.883	1:39:14
512	0.19	0.91	0.84	7:35:53
1024	0.220	0.893	0.811	35:19:46

Table 2: As Table 1 but up to 500 epochs with early stopping.

Further, we identify the receptive field as limiting factor for larger images. The receptive field of a filter is defined by the kernel size, stride, dilation, and padding of previous layers [6]. In the case of *U-net*, the receptive field at the deepest level grows exponentially with the number of encoder blocks: It has a side length of 14 pixels for one block, 32 for two blocks, 68 for three, 140 for four, 284 for five etc. In the next experiment, we adapt the number of *U-net* blocks for larger input sizes (see Table 3).

5 QUALITY META LABEL

Apart from the model, we also reconsider our dataset in order to improve results. Our dataset is heterogeneous. It contains images from various specimens and was collected by five researchers. Due to different experiment setups, they also vary in visual properties like brightness (standard deviation of per image average normalized brightness of 0.11), hue (0.22), saturation (0.12), and value (0.11) in *HSV* space. This affects the overall quality of each image. We empirically identify the following crucial factors: (1) contrast

Image Size	RF	CCE	mDSC	mIoU	Duration
256	$68_{b=3}$	0.149	0.934	0.881	**1:27:00**
512	$68_{b=3}$	0.144	0.934	0.879	9:30:14
512	$140_{b=4}$	**0.133**	**0.942**	**0.893**	8:47:55
1024	$68_{b=3}$	0.192	0.907	0.835	39:21:09
1024	$140_{b=4}$	0.179	0.920	0.856	22:45:53
1024	$284_{b=5}$	0.194	0.915	0.847	26:12:40

Table 3: Like Table 2, but increased receptive fields (RF). The subscript $_b$ denotes number of *U-net* blocks.

between the airway and the surrounding tissue (2) airway alignment in relation to the camera (3) sharpness of the airway edge (4) amount of airway occlusion (by tissue, particles or image distortions like flares). Because we found our perception influenced by these factors, we hypothesize this might be the case for our model. Based on these criteria, we manually categorize the samples into 22 "*bad*", 345 "*good*", and 53 "*great*" ones and perform further experiments.

First, we inspect data subsets by quality. A test set of 30 samples is randomly selected from the original dataset. It contains two "*bad*", 24 "*good*", and four "*great*" samples. The remainder is split by quality label. Each subset is used as training set and evaluated on the test set. The results are depicted in Table 4. The main difference in performance can be explained by different subset sizes: The best results are achieved when training on 321 "*good*" samples, while training on 49 "*great*" samples yielded the second best results, followed by training on 20 "*bad*" samples.

Next, we use random under sampling to balance the three subsets. When we repeat the experiment with only 20 samples per subset, the performance difference decreases (see $_{us}$ in Table 4). The prediction performance strongly correlates with our assigned quality label. We hypothesize that the order in which the samples are presented to the model might influence its performance. We test this hypothesis in an experiment, in which we use up to three distinct subsets to iteratively train a model. We stop training early and maintain the trained state of the model between sets. Because the order matters, we average the results of ten independent iterations instead of using cross validation as before. The results are depicted in Table 4. Overall, differences between settings are small, but there seems to be a slight advantage in starting with the worst and finishing with the best samples (bad →good →great).

6 APPROACH COMPARISON

When we consider the lessons learned in Sections 4 and 5, we create a model that differs from our defaults in the following way, in order to achieve the best trade-off between image resolution and prediction quality. (1) U-net with four encoder and decoder blocks (2) image dimensions of 512 by 512 pixels (3) training for 500 epochs with early stopping and a batch size of 8 (4) training by quality meta label (bad →good →great).

To compare our new approach to the semi-manual one, we use the test set defined in Section 5. The above U-net is trained on all other data, before being evaluated on the test data. An exemplary result is depicted in Figure 5.

Train Set and Order	CCE	mDSC	mIoU	Duration
bad	1.592	0.647	0.493	0:07:55
good	0.14	0.932	0.875	0:33:38
great	0.42	0.856	0.757	0:09:42
bad_{us}	1.404	0.651	0.498	**0:07:32**
good_{us}	1.156	0.748	0.616	0:18:28
great_{us}	0.553	0.836	0.726	0:08:18
bad →good →great	0.146	**0.945**	**0.898**	0:38:00
bad →great →good	0.14	0.934	0.879	0:38:53
good →bad →great	0.187	0.937	0.885	0:41:19
good →great →bad	0.216	0.933	0.878	0:37:20
great →bad →good	0.136	0.934	0.878	0:36:41
great →good →bad	0.228	0.934	0.881	0:37:32
good →great	0.168	0.941	0.891	0:32:03
great →good	0.129	0.936	0.881	0:33:59
all	**0.116**	0.941	0.889	0:30:51

Table 4: Experiment results for different data subsets by quality meta label and different order (Subscript $_{us}$ denotes a balanced (undersampling) version; set_a→set_b denotes that set_b was used for training after set_a)

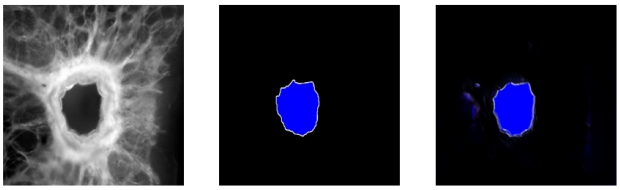


Figure 5: Sample input (left), target output (middle), and model output (right).

We let an expert use the old approach on the raw images of the test set. We then compare the semi-manual and the automated predictions with the labels created in Section 4.1 (Table 5). While our new approach yields slightly less accurate performance, it requires only a fraction of the time (mainly because it does not require human interaction).

Approach	mDSC	mIoU	Duration
IsoData	0.538	0.392	0:12:01
IsoData+	**0.966**	**0.935**	0:58:32
U-net	0.935	0.881	**0:00:44**

Table 5: Evaluation results for different approaches.

7 DISCUSSION AND CONCLUSION

In this work, we demonstrated how we transition from a semi-manual analysis of bronchodilators to an automated workflow using deep learning. To that end, we created a heterogeneous dataset. We presented how to use *Tensorflow* to create an optimized data pipeline; this pipeline can be used in other projects using *Tensorflow* with minor adjustments. Further, we propose an iterative model development process applicable to any data science project that requires custom code. We showed the capabilities of

our approach in a set of experiments for the semantic segmentation of airways. In these experiments, we trained a variant of *U-net* to segment airways in PCLS microscopy images with mIoU of 0.881 and mDSC of 0.935. We also demonstrated that image quality and training order can improve predictions. In comparison to our previous semi-manual approach, the proposed automated method yields comparable results, but does so in a significantly faster way. Thus, our automated approach constitutes a valid alternative. These segmentation results assign individual pixels to the airway target class. We can count those pixels to derive the airway area. From this, we can determine a relative change of the area (and hence the volume as described in Section 4.1) in a sequence of images and thus determine the bronchoconstriction.

Parts of our implementation are affected by randomness. As such, small differences might be due to chance. We expect additional improvements in semantic segmentation performance, once we add traditional or NN based data augmentation as in [23]. So far, we deliberately did not inspect this option to focus on the effect of other variables. The amplitude of differences reported in this work might decrease, once data augmentation is applied. Additionally, it should increase the robustness of our model. As the underlying data are image series, recurrent connections as in [2] could have the same effects. Further, we aim to extend our continuous integration pipeline by continuous deployment to further optimize our use of resources. There also exist other architectures that might be suitable for the task and require an in depth evaluation. For example, [3] proposes a fully CNN using dilated convolutions. Dilations allow them to increase receptive fields without increasing the kernel size, (i. e., covering a wider image area while maintaining the same amount of parameters).

ACKNOWLEDGMENTS

This work was supported by Fraunhofer Internal Programs under Grant No. Attract 042-601000 and Central Innovation Programme for small and medium-sized enterprises (ZIM) on behalf of Federal Ministry for Economic Affairs and Energy (BMWi) under Grant No. ZF4196502 CS7.

REFERENCES

[1] M. Abadi, P. Barham, J. Chen, Z. Chen, A. Davis, et al. 2016. *TensorFlow: A system for large-scale machine learning.* USENIX Association. 265–283 pages.

[2] M. Z. Alom, M. Hasan, C. Yakopcic, T. M. Taha, and V. K. Asari. 2018. Recurrent Residual Convolutional Neural Network based on U-Net (R2U-Net) for Medical Image Segmentation. (2 2018).

[3] M. Anthimopoulos, S. Christodoulidis, L. Ebner, T. Geiser, A. Christe, and S. Mougiakakou. 2019. Semantic Segmentation of Pathological Lung Tissue With Dilated Fully Convolutional Networks. *IEEE Journal of Biomedical and Health Informatics* 23, 2 (3 2019), 714–722.

[4] T. Araújo, G. Aresta, A. Galdran, P. Costa, A. M. Mendonça, and A. Campilho. 2018. Uolo - Automatic object detection and segmentation in biomedical images. In *Lecture Notes in Computer Science*, Vol. 11045 LNCS. Springer Verlag, 165–173.

[5] K. Beck. 2002. *Test Driven Development: By Example.* Technical Report. 240 pages.

[6] B. Behboodi, M. Fortin, C. J. Belasso, R. Brooks, and H. Rivaz. 2020. Receptive Field Size as a Key Design Parameter for Ultrasound Image Segmentation with U-Net. *Proceedings of the Annual International Conference of the IEEE Engineering in Medicine and Biology Society, EMBS* 2020-July (2020), 2117–2120.

[7] A. Buetti-Dinh, V. Galli, S. Bellenberg, O. Ilie, M. Herold, et al. 2019. Deep neural networks outperform human expert's capacity in characterizing bioleaching bacterial biofilm composition. *Biotechnology Reports* 22 (6 2019), e00321.

[8] S. W. D. Chien, S. Markidis, V. Olshevsky, Y. Bulatov, E. Laure, and J. Vetter. 2019. TensorFlow Doing HPC. *ieeexplore.ieee.org* (2019), 509–518.

[9] O. Danov, S. Jimenez Delgado, H. Drake, S. Schindler, O. Pfennig, C. Förster, A. Braun, and K. Sewald. 2014. Species comparison of interleukin-13 induced airway hyperreactivity in precision-cut lung slices. *Pneumologie* 68, 06 (6 2014), A1.

[10] A. Garcia-Garcia, S. Orts-Escolano, S. Oprea, V. Villena-Martinez, and J. Garcia-Rodriguez. 2017. *A Review on Deep Learning Techniques Applied to Semantic Segmentation.* Technical Report.

[11] A. Garcia-Uceda Juarez, R. Selvan, Z. Saghir, and M. de Bruijne. 2019. A Joint 3D UNet-Graph Neural Network-Based Method for Airway Segmentation from Chest CTs. In *Lecture Notes in Computer Science*, Vol. 11861 LNCS. Springer, 583–591.

[12] M. Havaei, A. Davy, D. Warde-Farley, A. Biard, A. Courville, et al. 2017. Brain tumor segmentation with Deep Neural Networks. *Medical Image Analysis* 35 (1 2017), 18–31.

[13] H. D. Held, C. Martin, and S. Uhlig. 1999. Characterization of airway and vascular responses in murine lungs. *British Journal of Pharmacology* 126, 5 (1999), 1191–1199.

[14] T. Kluyver, B. Ragan-Kelley, F. Pérez, B. Granger, M. Bussonnier, et al. 2016. Jupyter Notebooks—a publishing format for reproducible computational workflows. *Positioning and Power in Academic Publishing: Players, Agents and Agendas - Proceedings of the 20th International Conference on Electronic Publishing, ELPUB 2016* (2016), 87–90.

[15] S. A. Nadeem, E. A. Hoffman, and P. K. Saha. 2019. A fully automated CT-based airway segmentation algorithm using deep learning and topological leakage detection and branch augmentation approaches. In *Medical Imaging 2019: Image Processing*, Elsa D. Angelini and Bennett A. Landman (Eds.), Vol. 10949. SPIE, 11.

[16] V. Neuhaus, O. Danov, S. Konzok, H. Obernolte, S. Dehmel, et al. 2018. Assessment of the cytotoxic and immunomodulatory effects of substances in human precision-cut lung slices. *Journal of Visualized Experiments* 2018, 135 (5 2018), 57042.

[17] D. Paul, M. Steve, G. Andrew, and M. M. Stephen. 2007. *Continuous integration: improving software quality and reducing risk.* 336 pages.

[18] A. R. Ressmeyer, A. K. Larsson, E. Vollmer, S. E. Dahlèn, S. Uhlig, and C. Martin. 2006. Characterisation of guinea pig precision-cut lung slices: Comparison with human tissues. *European Respiratory Journal* 28, 3 (9 2006), 603–611.

[19] T. W. Ridler and S. Calvard. 1978. Picture Thresholding Using an Iterative Selection Method. *IEEE Transactions on Systems, Man and Cybernetics* SMC-8, 8 (1978), 630–632.

[20] O. Ronneberger, P. Fischer, and T. Brox. 2015. U-net: Convolutional networks for biomedical image segmentation. In *Lecture Notes in Computer Science*, Vol. 9351. Springer Verlag, 234–241.

[21] E. Shelhamer, J. Long, and T. Darrell. 2017. *Fully Convolutional Networks for Semantic Segmentation.* Technical Report 4. 640–651 pages.

[22] J. Vietmeier, F. Niedorf, W. Bäumer, C. Martin, E. Deegen, B. Ohnesorge, and M. Kietzmann. 2007. Reactivity of equine airways - A study on precision-cut lung slices. *Veterinary Research Communications* 31, 5 (7 2007), 611–619.

[23] J. Wang and L. Perez. 2017. *The effectiveness of data augmentation in image classification using deep learning.* Technical Report.

[24] A. Wohlsen, C. Martin, E. Vollmer, D. Branscheid, H. Magnussen, et al. 2003. The early allergic response in small airways of human precision-cut lung slices. *European Respiratory Journal* 21, 6 (6 2003), 1024–1032.

[25] N. Wu, J. Phang, J. Park, Y. Shen, Z. Huang, et al. 2019. Deep Neural Networks Improve Radiologists' Performance in Breast Cancer Screening. *IEEE Transactions on Medical Imaging* (10 2019), 1–1.

[26] Bo Zhao, Jiashi Feng, Xiao Wu, and Shuicheng Yan. 2017. A survey on deep learning-based fine-grained object classification and semantic segmentation. *International Journal of Automation and Computing* 14, 2 (2017), 119–135.