ulm university universität
uulm

Ulm University | 89069 Ulm | Germany

Faculty of Engineering,
Computer Science and Psychology
Institute of Media Informatics
Visual Computing Group

Convolutional Neural Networks (CNN) Applied to Respiratory Motion Detection in Fluoroscopic Frames

Master Thesis in Computer Science at Ulm University

Presented by:
Christoph Baldauf
christoph.baldauf@uni-ulm.de

Examiner:
Prof. Dr. rer. nat. habil. Timo Ropinski
Prof. Dr. rer. nat. Volker Rasche

Advisor:
Alex Bäuerle

2019/01

Last updated January 10, 2019

Typesetting: PDF-LaTeX 2$_\varepsilon$

Abstract

In cases of severe aortic stenosis, the aortic valve can be replaced within a catheter intervention that is performed under X-ray-fluoroscopy guidance. X-ray has difficulties in imaging soft tissues since attenuation of the radiation between them is very similar. This weakness can be addressed by augmenting real-time fluoroscopy with organ shape models derived from e.g. pre-interventionally acquired CT angiography (CTA). Following initial registration, respiratory motion is a major cause of introducing mismatch to the superposition. If this motion can be extracted from the fluoroscopy, the model overlay position can be adjusted accordingly, and the mismatch can be reduced.

This work evaluates convolutional neural networks (CNN) as a novel approach to address the problem. Respiratory motion correlates well with the diaphragm position, which is often the most prominent anatomic structure in the fluoroscopic frames. Thus the CNN is trained on fluoroscopic frame pairs as input with the target value being the displacement of the diaphragm between these frames. The displacement is restricted in head-feet direction only and is determined by defining a region of interest (ROI) on the diaphragm edge and tracking its motion with cross-correlation using Matlab.

Training data were derived from transcatheter aortic valve implantation (TAVI) procedures performed at the Ulm University Medical Center.

Implementation was done in Keras, using TensorFlow as back-end. All implemented network configurations were tested and evaluated by calculating error rates on the validation and test set. Furthermore, attention heatmaps were used to identify which regions of the input are particularly important to the network.

Convolutional neural networks have shown to be capable to extract respiratory motion from fluoroscopic frames. Quality is sufficient to reliably detect respiratory phase and rate. Deviations from the target values of the dataset were observed especially for large respiratory amplitudes. Improvements of the dataset quality might lead to a higher accuracy.

Contents

1. Introduction **1**
- 1.1. Problem Context . . . 1
- 1.2. TAVI . . . 1
- 1.3. Motivation . . . 3
- 1.4. Research Question . . . 3
- 1.5. Overview . . . 4

2. Related Work **5**
- 2.1. TAVI . . . 5
- 2.2. Detection of Respiratory Phase . . . 5
- 2.3. Deep Learning in Radiology . . . 6
- 2.4. Deep Learning for Motion Detection . . . 7
- 2.5. Conclusion . . . 8

3. Data Pipeline **9**
- 3.1. Data Acquisition . . . 10
- 3.2. Data Preprocessing . . . 10
- 3.3. Template Matching . . . 10
- 3.4. Tracking the Diaphragm with Matlab . . . 12
- 3.5. Generation of a Lookup Table . . . 14
- 3.6. Final Dataset . . . 16

4. Implementation **19**
- 4.1. Convolutional Neural Networks . . . 19
  - 4.1.1. Layers . . . 20
  - 4.1.2. Training . . . 22
- 4.2. Choice of Proper Network Type . . . 22
- 4.3. Choice of Foundational Network Architecture . . . 23
- 4.4. Libraries and Hardware . . . 24
  - 4.4.1. Libraries . . . 24
  - 4.4.2. Hardware . . . 25
- 4.5. Keras Implementation . . . 25
  - 4.5.1. Project Structure . . . 25

4.5.2. Validation Method 27
4.5.3. Unoptimized Network 27
4.5.4. Dummy Dataset 30
4.5.5. Hyperparameter Optimization 32
4.5.6. Dataset Modification 33
4.5.7. Siamese Network 35

5. Evaluation **37**
5.1. Training Results with Final Dataset 37
5.2. Heatmap Plots 39
5.3. Data without Diaphragm 42
5.4. Drawbacks of Dataset Quality 44
5.5. Interpretation 47

6. Conclusion **49**

A. Appendix **51**
A.1. Program Code 51
A.1.1. Network Implementation 51
A.1.2. Generator Module 53

1. Introduction

1.1. Problem Context

The cardiovascular system permits blood to circulate and transport nutrients from and to cells in the body. The heart is one key part in this system, which pumps oxygenated blood to the body and deoxygenated blood to the lungs. In the human heart there is one atrium and one ventricle for each circulation and therefore four chambers in total: left atrium, left ventricle, right atrium and right ventricle.

Blood flow is only allowed in one direction in the human heart. This is ensured by four valves. The valve between left ventricle and aorta is called aortic valve. Malfunction of it is often caused by a condition called aortic stenosis, which is a narrowing of the valve. It requires the heart to use an increased force to pump blood into the artery. The clinical manifestation consists of three main symptoms: angina, cardiovascular syncopes and heart failure [49]. Aortic stenosis is mostly caused by an age-related calcification with a prevalence of only about 0.2% in the age group 50-59 but increases to almost 10% for adults over 80 years [8]. Especially in the context of a rapidly aging population, aortic stenosis, with its severe complications, is a major cardiac concern and also an economic burden.

Within this thesis, we aim to use artificial neural networks to improve a treatment option for aortic stenosis.

1.2. TAVI

The two main treatment options are a conventional aortic valve replacement (AVR) or transcatheter aortic valve implantation (TAVI) [49]. The latter is a minimally invasive treatment for patients who are not eligible to undergo conventional surgery (e.g. due to comorbidities, pregnancy, etc.) and is, for instance, performed at the Ulm University Medical Center.

During the intervention, a catheter is introduced into the femoral artery in the groin and moved up to the heart where it replaces the damaged valve. Knowledge about the exact position of the catheter is crucial for the success of the performance since malposition

can result in severe complications [24]. For this reason, the catheter's path is guided with the help of real-time X-ray fluoroscopy superimposed with preoperatively acquired computed tomography angiography (CTA) models. This image fusion allows utilizing the high resolution and real-time capabilities of X-ray while compensating its low soft-tissue contrast. Figure 1.1 shows an aorta model superimposed onto a fluoroscopic frame.

Figure 1.1.: Superimposition of an aorta model (red) within a fluoroscopic frame. The model is static and acquired preoperatively, while fluoroscopic frames are acquired in real-time during the intervention. Through the superimposition, structures like the aorta are clearly visible and this augmented view can be used to guide a catheter to the correct location for replacing the aortic valve since the location within the vessel is known. The balloon-expandable prosthetic valve is highlighted in the image.

1.3. Motivation

One limiting factor of this approach is overlay misalignment caused by respiratory motion [44][43]. If the patient breathes, organs in the chest are in motion and the location and shape of soft tissue structures (e.g. heart, blood vessels, etc.) change. In contrast, the CTA-model remains at the same location and therefore the superimposition loses accuracy. There are two options for motion compensation:

1. Extraction of respiratory displacement and using this information to shift the CTA-model accordingly.
2. Determination of the respiratory phase and then superimposing a matching CTA-model from a set of CTA-models acquired in different respiratory and cardiac phases.

This thesis aims to extract respiratory displacements, i.e. option 1. The current approach is to select the diaphragm, a prominent shape in the fluoroscopic frame whose motion primarily correlates with respiratory motion and track its position with template matching technologies such as cross-correlation.
This approach has shortcomings: it requires manual selection of a region of interest (ROI) to track and if the diaphragm has no clear contour, these algorithms fail. Better algorithms for this problem can improve the accuracy of the image-guided TAVI procedure.

1.4. Research Question

Neural networks have become a powerful technology for image related tasks. This thesis evaluates convolutional neural networks (CNN) as new approach to detect and quantify respiratory motion. The network is trained on fluoroscopic frame pairs that are acquired during the TAVI procedures at the Ulm University Medical Center. The displacement of the diaphragm between these two frames is used as target value and is calculated through cross-correlation. The implementation is done using the TensorFlow-framework [1] as back-end.
The three main contributions of this work are:

1. evaluating network architectures to detect displacements of the diaphragm,
2. evaluating the network's performance on the problem, especially in comparison to cross-correlation,
3. evaluating the network's performance on frame pairs that do not contain the diaphragm and therefore make a cross-correlation not applicable.

1.5. Overview

In Chapter 2, we discuss work that is related to this thesis. Topics that are covered include TAVI, respiratory motion detection, motion detection with neural networks and the application of neural networks in the field of radiology. In Chapter 3 we discuss the steps involved in creating the dataset. This starts with acquiring X-ray fluoroscopy runs during the TAVI intervention and leads up to the labeled dataset.
In Chapter 4 we move to the neural network part. We discuss which architecture and configuration were used as starting point, which steps were involved in optimizing the network and how validation of the network was done. Chapter 5 evaluates the optimized network with respect to the validation and test set and discusses limitations. Finally, Chapter 6 discusses to what extent the research question has been answered and which potential improvements can be made in future work.

2. Related Work

This chapter discusses important basics for the subject of this thesis. The basics can be subdivided into four parts. The first part deals with the medical aspects and gives an overview of the TAVI procedures. The second part focuses on the application of deep learning technologies in radiology. Subsequently, neural networks for motion detection are presented. Finally, non-learning approaches for the detection of respiratory motion are presented.

2.1. TAVI

The first transaortic valve implementation (TAVI) was described by Andersen *et al.* in 1998 by presenting results of the implementation in pigs [5]. In 2002, the first TAVI procedure was performed in humans [12]. While some of the first patients died shortly after the intervention due to severe comorbidities, other elderly patients lived up to 6.5 years with the new valve. These first trials proved the feasibility of the intervention. Over the years, transcatheter devices and valve types further improved and the intervention became common at many locations, including the Ulm University Medical Center.
Recent developments in intraprocedural image fusion addressed the limited soft tissue contrast in X-ray with help of echocardiography [6] or computed tomography angiography (CTA) [37]. 3D-CTA enhanced X-ray fluoroscopy was evaluated in Ulm with a focus on guiding double-filter embolic protection devices [43] and the reduction of necessary radiation doses [44].
Within this work, we try to further improve imaging technologies to support TAVI procedures.

2.2. Detection of Respiratory Phase

Numerous work has been published with respect to respiratory phase detection, however, instead of detecting the respiratory waveform, many works focus on only differentiating between the phases inspiration and expiration.

For the latter, an overview is given by Palaniappan *et al.* in [33]. They distinguish between two approaches. An acoustic approach uses breath sound signals, e.g. originating from the trachea, and analyzes them in time- or frequency domain. Most algorithms assessed can detect inspiration and expiration with accuracy well above 95%. Error source is mostly due to swallowing. The paper does not evaluate how powerful algorithms are for measuring the exact respiratory waveform, but they seem to provide poor performance on this application.

This can be exemplarily seen in one work of Chuah and Moussavi [19] that uses tracheal and chest sound signals to detect respiratory phases. While inspiration and expiration are detected reliably, airflow is additionally estimated based on the sound signal and compared with the actual airflow and plotted in a graph. The accuracy in airflow estimation is worse than the 93% reached in the detection of inspiration or expiration, as can be seen by visually analyzing Figure 1. Presuming, lung capacity is known, airflow can be used to determine the respiratory waveform. However, considering the results of this paper, this does not seem to be very promising.

The second approach is called non-acoustic and is based on sensors. One common technique is called respiratory inductance plethysmography (RIP) and was introduced by Konno and Mead in 1967 [21]. The system consists of two wire coils encircling the rib cage and the abdomen. During respiration, the self-inductance of the coils changes and alters the frequency of an attached oscillator. After calibration with the help of a spirometer, a volume-motion relationship can be established with the abdominal- and rip cage displacement which is approximately linear. Performance of this technique was evaluated by Carry [9]. The mean deviation between RIP and a pneumotachography is 4.6% and therefore seems reasonably accurate for analysis of respiratory waveform. Error occurs primarily during the start of inspiration and expiration (underestimated volume) and end of inspiration and expiration (overestimated volume).

Our approach would improve the detection of the respiratory phase since it does not require any sensors to be mounted on the patient. Information about respiratory motion is available within the fluoroscopic frames and if it can be extracted, the quality of the image guidance can be improved without requiring any additional effort from the medical staff or causing higher radiation doses. Furthermore, there are no costs for additional necessary equipment.

2.3. Deep Learning in Radiology

Convolutional neural networks were introduced by Yann Lecun in 1998 [27] and over the last 5 years became a powerful technology for speech- and image related tasks. In the medical field, a focus of research lies in radiology due to a large amount of data

available. Mazurowski *et al.* give an overview of deep learning in radiology in [31]. They identify three main applications:

- **Classification** of diseases in medical image data (e.g. tuberculosis)
- **Segmentation** of an image in order to separate distinct parts/objects
- **Detection** of objects within an image

Other applications on images include registration, enhancement and quality assessment. In classification tasks, neural networks can reach better results than visual inspection by humans. One example is a network called *CheXNet* proposed by Rajpurkar *et al.* in 2017 [35] that detects pneumonia from chest X-rays more reliably than practicing radiologists.

Most of the medical applications for neural networks tackle one specific problem. For fluoroscopic frames, available work deals, for instance, with image enhancement [42] and catheter segmentation [2]. Our domain was to improve respiratory motion compensation with such networks. With respect to the aforementioned works, there are differences. First, our motion quantification requires to process more than one frame at a time, which is not the case in the cited works. Second, we not only detect objects but also track their displacement, which is an additional task. This means that the problem is twofold: Objects that are suitable to measure respiratory motion must be located and also their displacement with respect to other frames must be quantified by the same neural network.

2.4. Deep Learning for Motion Detection

Zhang *et al.* evaluated different network types for the detection of vehicle motion [47]. Training- and validation are taken from the KITTI dataset [15] which is captured while driving in the Karlsruhe area. As network type, they evaluated ResNet, AlexNet and a CNN consisting of 2 convolutional/pooling- and 2 fully connected layers. The latter showed the best performance, however, the validity of the results is limited, since the small dataset might be insufficient for complex networks such as ResNet and AlexNet. Furthermore, this work differs from our approach since the input to the neural network consists of one rgb-frame of the street scenery and the corresponding optical flow.

Two other works deal with optical flow estimation. FlowNet [13] is proposed by Dosovitskiy *et al.* The network is relatively deep (9 convolutional layers) and achieves good results on common optical flow benchmarks (Sintel and KITTI). It takes two images as input and calculates the optical flow between them. They evaluate two architectures: One network is a rather generic CNN that takes a two-channel image as input. The other is a siamese network that processes the input images in two separate streams

before merging the results and further processing them to get an output. Depending on the dataset, one or the other architecture yielded better results, but the difference in accuracy is rather small. Concurrently with that, Teney and Herbert introduced an architecture derived from signal processing principles that contain fewer weights and therefore requires less training data [40]. With only two convolutional layers the network is shallow and according to the paper, this results in significantly worse performance on large displacements compared to FlowNet. For smaller displacements results are comparable.

Complementary to these works, our approach predicts the displacement of the diaphragm, which is the motion of one specific shape based on two fluoroscopic frames. This differs to the optical flow prediction since in this case motion in all regions of the image are analyzed and the network training input are optical flow maps. The target value (vehicle speed) of the vehicle motion prediction is similar to our problem, however, the network uses optical flow maps as training input, which is significantly different to our approach.

2.5. Conclusion

Related work has shown that neural networks are a powerful technology for all sorts of image-related tasks. They have been used numerously in the field of radiology and have also been assessed for motion detection. However, works dealing with motion detection are using optical flow maps for training, which requires higher effort to prepare a dataset and might be not possible to be done with noisy fluoroscopic frames. Additionally, regression problems are comparatively little subject of research and the detection of respiratory motion has not been addressed with neural networks in research before. Since information about respiration is visible within fluoroscopy run our approach seems to be promising and has the advantage of not requiring any additional equipment or effort from side of the medical staff. Evaluating this application and highlighting potentials for further research make the subject of this thesis worth to be evaluated.

3. Data Pipeline

An intuitive approach to address the problem of identifying the displacement of the diaphragm is to take two frames of a fluoroscopy run as input for a neural network and train it to predict the respiratory displacement in pixels. Therefore, one training/validation sample consists of two frames and a scalar value as target value. The latter is obtained by tracking the motion of the diaphragm, which is the main muscle for respiration [36]. Due to its prominent appearance within the fluoroscopy run, tracking algorithms achieve reliable results. In this work, it was assumed that respiratory motion only takes place in the head-feet-direction.

Figure 3.1 illustrates two fluoroscopic frames of the same run and the displacement as determined via the diaphragm.

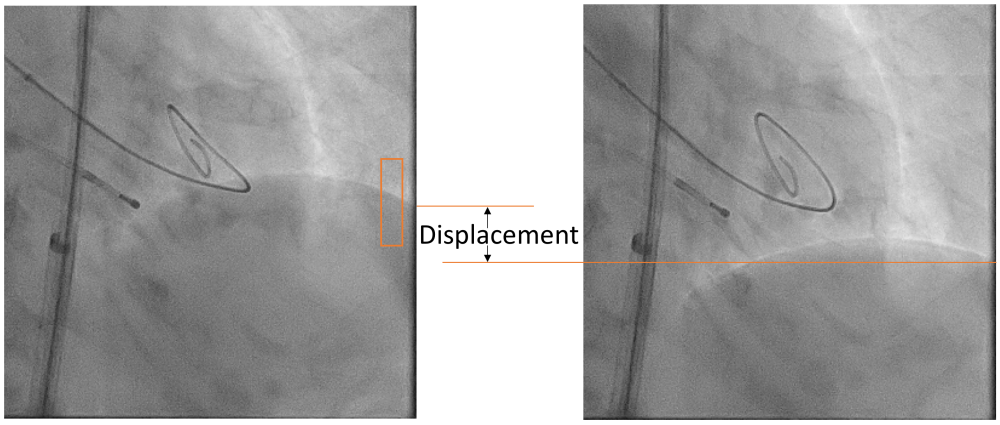


Figure 3.1.: First frame of fluoroscopy run (left) and 30th frame (right). Displacement, as measured by the edge of the diaphragm, is highlighted and is 58 pixels in this example. The rectangular box in the left frame highlight the selected ROI

The data pipeline from image acquisition to an applicable dataset for a neural network will be described in the following sections of this chapter.

3.1. Data Acquisition

TAVI procedures are performed in one of the cardiac catheterization labs at the university hospital. Each of the labs is equipped with an Allura Xper FD10 cardiovascular X-ray system (Philips Healthcare, Best, The Netherlands). The camera angle of the projection can be adjusted arbitrarily by the interventional operators in both CRA/CAU and RAO/LAO direction. Also, the distance of the X-ray tube to the patient can vary. The resulting 8-bit grayscale frames have a resolution of 512 x 512 pixels and are sampled at 15 frames/second.

Two of the labs are additionally equipped with image-guidance systems (EP Navigator, Philips Healthcare, Best, The Netherlands) that overlay 3D cardiac CT data onto X-ray fluoroscopy. These systems could reach a higher accuracy if information about respiratory motion is available to them, e.g. through a neural network.

Several fluoroscopy runs are generated during the intervention and stored as DICOM files that can be later obtained from the clinical image storage system (PACS). Due to restrictions, direct access to the data from a computer is not possible, therefore the relevant data were transferred via DVD.

3.2. Data Preprocessing

All data were viewed with the DICOM viewer Horos [32] and files that do not contain the diaphragm were sorted out (with some exceptions that were used as test data) since labeling them for use as training data is very difficult due to the lack of a prominent shape that can be tracked. The remaining files were converted to lossless grayscale AVI videos with a Matlab [30] script. This allows to further process the data within the Matlab computer vision toolbox.

3.3. Template Matching

The next step in the data pipeline is to track the motion of the diaphragm. There are numerous tracking algorithms available. Initially, the Lucas-Kanade-Algorithm [29] was evaluated, however, simple template matching by cross-correlation, as described in the following, yielded better results.

Template matching is a technique in digital image processing for finding parts of an image that match a template image. The matching process moves the template to all possible positions in a larger source image and computes a numerical index that indicates how well the template matches the image in that position [25]. An illustration of the method is shown in Figure 3.2. The calculation is done on a pixel-by-pixel basis

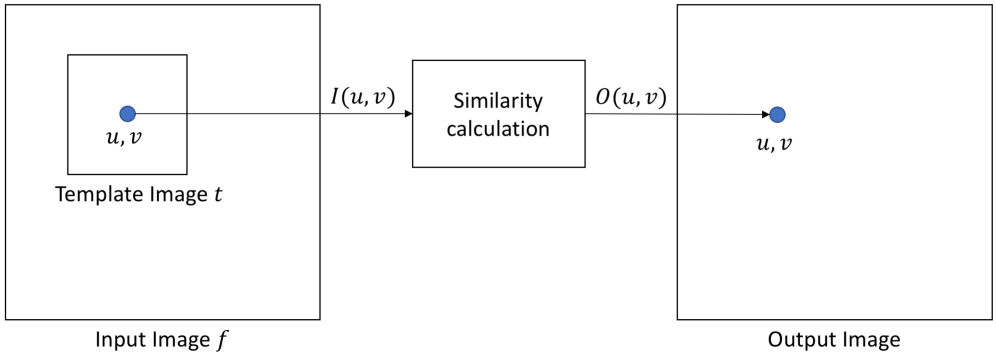


Figure 3.2.: Functionality of template matching. The template image t is centered to all positions in the image f. A scalar value indicating similarity is calculated based on the template image and the image under the region of the template. This scalar value is written to the corresponding position in an output image.

based on the template at position (u, v) and the input image under the region of the template. A common similarity measure is the Euclidian distance:

$$d^2_{f,t} = \sum_{x,y} [f(x, y) - t(x - u, y - v)]^2 \tag{3.1}$$

where the sum is calculated over the region of the template at the position u, v. This term can be expanded which yields in:

$$d^2_{f,t} = \sum_{x,y} [f^2(x, y) - 2f(x, y)t(x - u, y - v) + t(x - u, y - v)^2] \tag{3.2}$$

Under the assumption that $\sum f^2(x, y)$ is approximately constant and by discarding the constant term $\sum t(x - u, y - v)^2$ the cross-correlation is calculated by

$$c(u, v) = \sum_{x,y} f(x, y)t(x - u, y - v) \tag{3.3}$$

Using equation 3.3 as a measure for similarity has several disadvantages [28].

- If the energy of the image $\sum f^2(x, y)$ varies across the image, matching can fail. For instance, a bright spot can yield a higher correlation than an exactly matching dark part of the image.

- The range of $c(u, v)$ depends on the size of the template.

- Changes in image amplitude due to changing lighting conditions across the image sequence yields different results.

We try to address these disadvantages by using the local sums to normalize the cross-correlation. The normalized cross-correlation can be calculated by

$$\gamma(u,v) = \frac{\sum_{x,y}[f(x,y) - \bar{f}_{u,v}][t(x-u,y-v) - \bar{t}]}{\sqrt{\sum_{x,y}[f(x,y) - \bar{f}_{u,v}]^2 \sum_{x,y}[t(x-u,y-v) - \bar{t}]^2}} \tag{3.4}$$

where f is the image, $\bar{t}$ the mean of the template and $\bar{f}_{u,v}$ the mean of $f(x,y)$ in the region under the template. The correlation coefficient calculated with equation 3.4 ranges from -1 to 1, with 1 indicating maximum similarity and -1 maximum deviation.

3.4. Tracking the Diaphragm with Matlab

The built-in function *normxcorr2()* calculates the cross-correlation as described in the previous subsection. It centers the template to all positions of the image. The image is padded with zeros at the borders. An image f of size $(w_f \text{ x } h_f)$ correlated with a template t of size $(w_t \text{ x } h_t)$ results in a matrix of correlation coefficients of size $(w_f + w_t \text{ x } h_f + h_t)$. An example is illustrated in Figure 3.3.

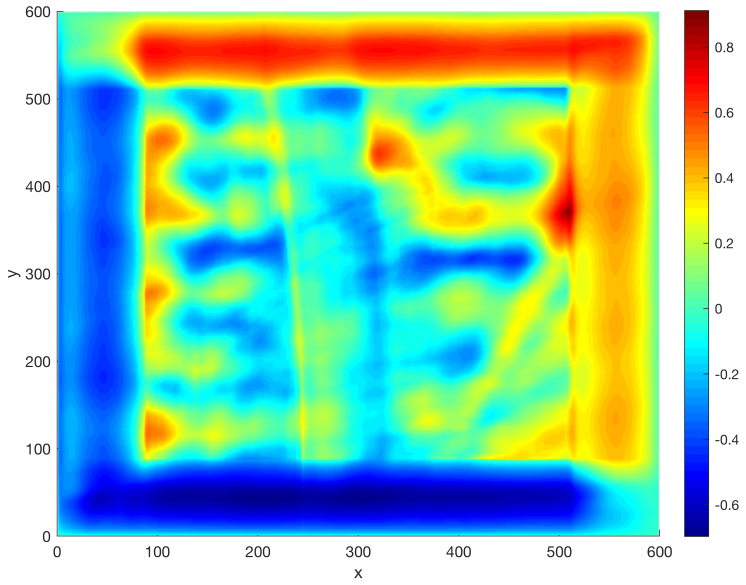


Figure 3.3.: Heatmap of correlation coefficients with a maximum match at x = 509 and y = 374.

The tracking of the diaphragm is done with a Matlab script. In a first step, one fluoroscopy run is loaded and the first frame is shown in a figure. Selection of a ROI, i.e. the template to be matched is done by drawing a rectangular box in the frame (see Figure 3.4).

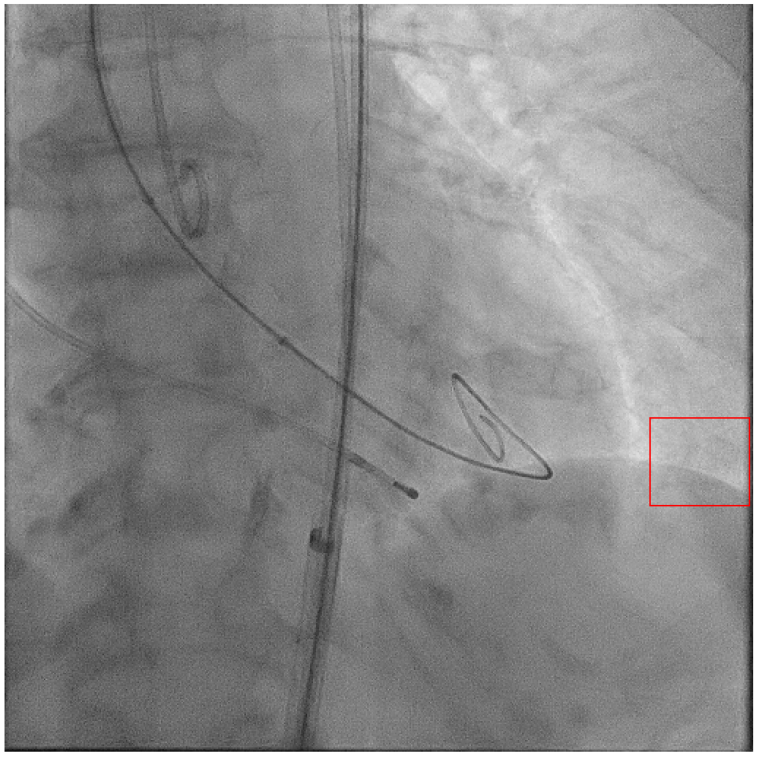


Figure 3.4.: First frame of fluoroscopy run and selected ROI (red frame). The edge of the diaphragm is usually visible in the bottom right of a frame.

The ROI is selected such that it contains the edge of the diaphragm. Following this, the best match of this template is determined in all consecutive frames and the results are smoothed by a third-order median filter. Occasionally, matching fails which can be seen by large variations in the displacement. Then, trying a slightly different ROI finally often yields the desired result. Plausibility is evaluated by visually comparing the fluoroscopy run with the displacement graph. Figure 3.5 shows the displacement of the diaphragm over a fluoroscopy run.
Additionally, the script creates a folder for every frame in the fluoroscopy run that contains three files:

1. *1.png* – Reference frame (first frame of fluoroscopy run)
2. *2.png* – Current frame of the fluoroscopy run

3. *distance.txt* – Textfile containing the displacement between the frames

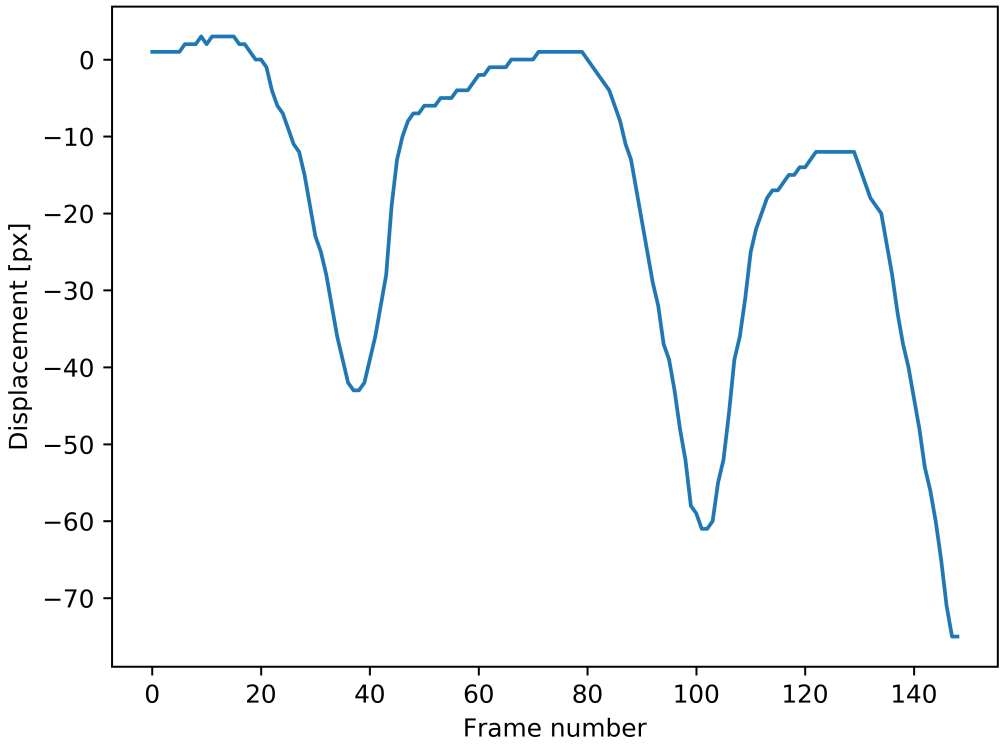


Figure 3.5.: Respiratory motion in fluoroscopy run. In this graph, a negative value corresponds to expiration. Since the displacement merely becomes positive, the reference frame is recorded while the patient is in almost full inspiration.

3.5. Generation of a Lookup Table

The initial idea was to use the folders as input to the neural network since one folder contains one sample and most neural network frameworks provide functions to use folders as input. This had provided only a limited number of samples because the displacement of each frame is only used with respect to the always same reference frame (first frame of fluoroscopy run). The number of samples can be vastly increased by analyzing all possible combinations of frames within a fluoroscopy run, which makes the previous approach of using folders for training impractical.

Amidi *et al.* propose a convolutional neural network that is used for enzyme classification [3]. They posted a blog article about how they pass a large amount of data into the neural network and made the code free to use [4]. This code can be easily adjusted

for the data in this thesis and allows to generate data samples in real time on multiple cores and feed it into the neural network. The adopted idea works as follows:

- Let **ID** be a python string that identifies a sample within the dataset
- Create a dictionary called **partition** with two entries:
  - **partition['train']** gathers a list of all training IDs
  - **partition['validation']** gathers a list of all validation IDs
- Create a dictionary called **data_set** where for each ID of the dataset, the associated images and labels are given by **data_set[ID]**

For example, if the dataset contains the training ID **id-1** and **id-2** and the validation ID **id-3**, partition looks like

```
>>> partition
{'train': ['id-1', 'id-2'], 'validation': ['id-3']}
```

and

```
>>> data_set
{'id-1': ['path-to-image-1, 'path-to-image-2, 'displacement']
'id-2: ['path-to-image-3, 'path-to-image-4, 'displacement']
'id-3: ['path-to-image-4, 'path-to-image-5, 'displacement'] }
```

These dictionaries are created with a Python script that iterates through all fluoroscopy runs and creates all possible combinations of frames. The number of possible combinations can be calculated with equation 3.5 where n is the number of frames in a fluoroscopy run. Since the input order of the images to the neural network can be swapped, the number of combinations is multiplied by 2.

$$number\_combinations = 2 * \binom{n}{2} = 2 * \sum_{i=1}^{n-1}(i) \tag{3.5}$$

One issue with this approach is that long fluoroscopy runs deliver an overproportionally large number of samples. For instance, a run with 50 frames provides 2,450 and a run with 300 frames 89,700 samples. In order to prevent that the vast majority of the dataset is composed of few explicitly long runs, only the first 110 frames of each run are used. The value of the displacement is scaled to a range between 0 and 1. This is done by finding out the absolute values of the largest and smallest displacement within all samples. After adding both, this value is used as upper bound and the negative same value as lower bound for the range that is mapped into a range between 0 and 1. Everything in between these bounds is scaled linearly.
The dataset is split into three categories:

Training Set: Data used to train the neural network

Validation Set: Data used to evaluate the performance of the network while tuning hyperparameters

Test Set: Data used to evaluate the performance of the final model. Ensures that the model is not overfitted to the validation set and fails on other data

The ratio between these sets is roughly 85% training, 8% validation and 7% test set. Data (fluoroscopy runs) are assigned to the groups randomly. A property of the test set is that the approach of creating all possible combinations is not applied to it. In the test set each frame is assigned a displacement only with respect to the first frame as reference. This allows to predict the displacement over the course of the actual fluoroscopy run and overlay the results on the displacement graph created with Matlab. This gives a visual impression of the network's performance while in contrast, the performance on the validation set is evaluated by calculating the error on all possible combinations.

3.6. Final Dataset

The dictionaries **partition** and **data_set** are stored as NumPy file and can be loaded and used for the neural network. The composition of training, validation and test set can be seen in Table 3.1. Distribution of samples with respect to the displacement is illustrated in Figure 3.6. It can be seen that the distribution is very imbalanced and concentrated at small displacements. This could be an issue since a network might overfit to small values. An additional Python script allows modifying the distribution by either setting a cap, i.e. only use n samples per displacement or replicating samples which are underrepresented.

	No. of patients	No. of fluoroscopy runs	No. of samples
Training Set	69	131	728,824
Validation Set	10	10	78,884
Test Set	9	9	1,252
Total	88	150	808,960

Table 3.1.: Composition of data_set. Most of the available fluoroscopy runs are used for training, while roughly 15% are in the validation or test set.

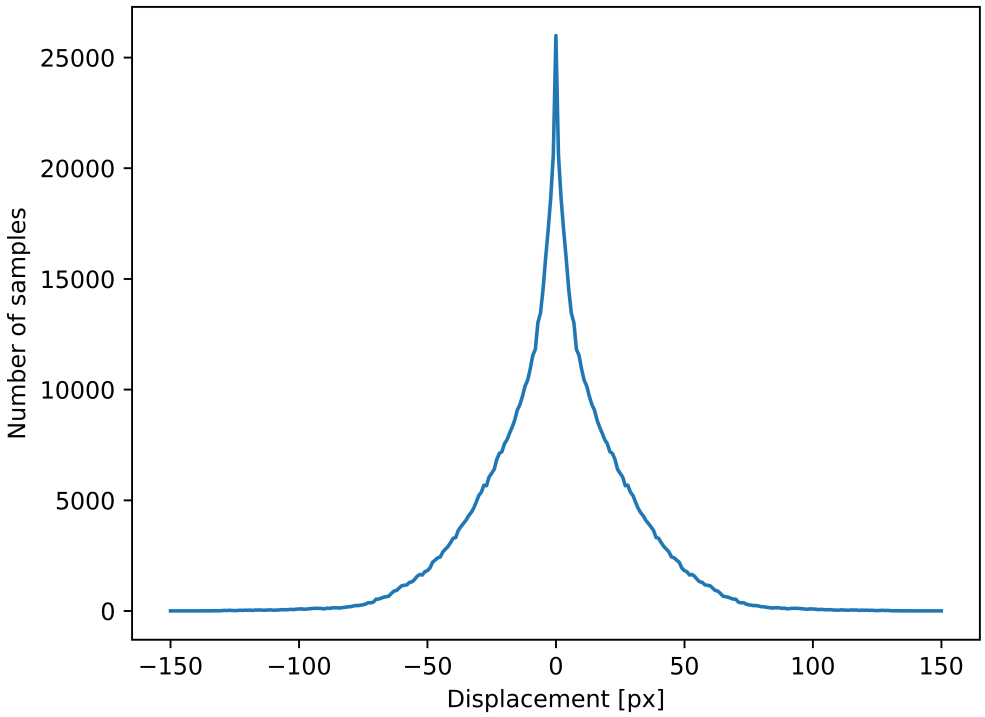


Figure 3.6.: Histogram of samples per displacement of the dataset. Small displacements compose the gross of the dataset, while samples with a displacement over absolute 60 pixels occur rarely. The histogram is symmetric across the y-axis.

4. Implementation

This chapter describes the network implementations used to approach the problem. Initially, convolutional neural networks as a variant of artificial neural networks and the motivation for the initial unoptimized architecture are described. This is followed by a listing of used libraries for building neural networks and an overview of the project structure, i.e. the software files created. The remainder describes the initial network, the approach to optimize it and test cases used to validate the network's performance.

4.1. Convolutional Neural Networks

Convolutional neural networks (CNN) are a subtype of artificial neural networks. As they are also made of neurons that have learnable weights and biases, they are very similar to regular artificial neural networks. An issue with these is that they do not scale well and even modestly sized images have an enormous amount of information. A fluoroscopic 512x512 frame contains 262,244 pixels. Each neuron in the regular hidden layer, therefore, adds 262,244 weights. Even for small networks, this would lead to a huge number of trainable parameters which is not computationally feasible.
Furthermore, many applications require to be translation invariant. For regular neural networks, this means that they had to be trained to detect a pattern at different locations in an image [26].
CNNs encode some properties that allow overcoming these problems. They adopt a biological concept called the receptive field. A receptive field is a restricted area in the visual field of an animal that is processed by a neuron. These neurons in the visual cortex act as detector for certain types of stimuli, for instance, edges. Receptive fields are found all across the visual field and also overlap each other.

4.1.1. Layers

In the following subsections, we will the explain the basic layer types (convolutional-, pooling- and fully connected layer) we used in our network architecture.

Convolutional Layer

The functionality of receptive fields is approximated by using convolution operations with small filter sizes (e.g. 3 x 3). This property is called *local connectivity* since every neuron is connected to a local region of the input (region under the filter). The connections are local in space, but full along the entire depth of the input volume.
Images can be filtered to produce different visible effects [17, p. 327]. An example of a filter operation is shown in Figure 4.1. The Sobel operator is convolved with the original image and calculates the derivative in y-direction and creates an image that emphasizes edges in that direction.

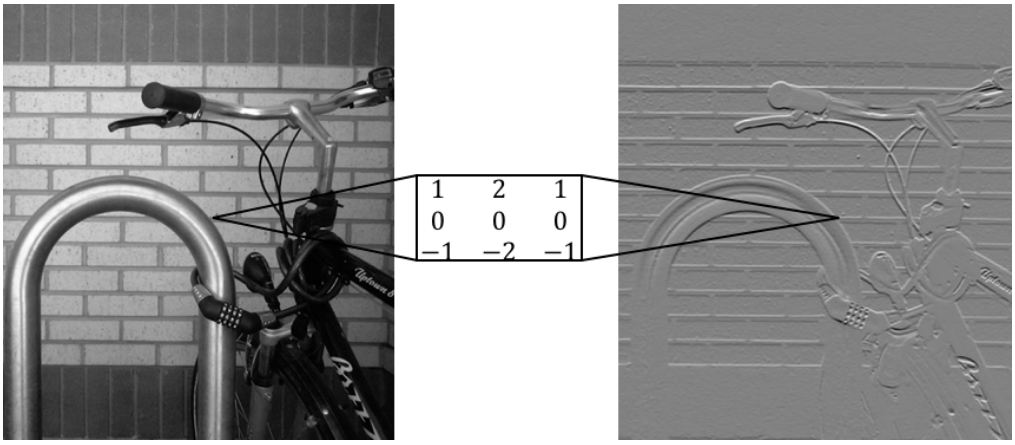


Figure 4.1.: The dot product of the filter with every subimage is calculated and creates an output image, also called feature map. Images are taken from [46].

Multiple convolutional filters compose a convolutional layer of a neural network. The matrix values of the filters are parameters that are not predefined, in contrast to the Sobel operator, but trained by using machine learning. The convolutional operation replaces the multiplications in regular neural networks [17, p. 330]. Output of the layer is a volume, with width and height depending on the size of the input to the layer and the parameters of it. The depth depends on the number of filters.
Since each neuron in a depth slice of the volume uses the same filter, the number of parameters is drastically reduced. This property is called *parameter sharing* and ensures translation invariance [17, p. 335]. Multiple convolutional layers form a convolutional neural network. The idea is that layers close to the input recognize low-level features,

such as edges and corners. These features are combined to high-level features, such as faces, in layers closer to the output.

Pooling Layer

Another concept of CNNs is called pooling, which is a form of downsampling [17, pp. 335-339]. It is common to periodically insert such layers between successive convolutional layers and it replaces the output of a preceding one with a summary statistic of nearby activations.
A common type is called max-pooling, in which the feature map is split into depth slices and each slice is divided into rectangular regions. Of each region only the maximum value is passed on. The size of this region is usually 2x2 pixels with a stride of 2 and therefore reduces the resolution by a factor of four (in 2D). The depth of the resulting feature map remains unchanged. An illustration of max-pooling is shown in Figure 4.2.

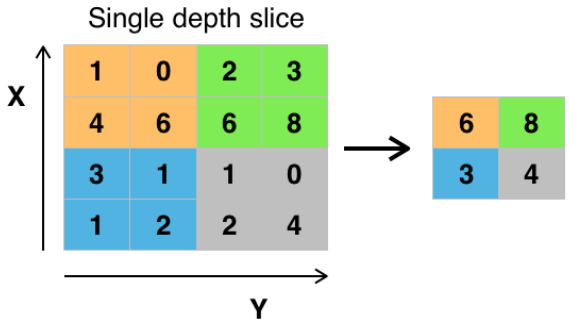


Figure 4.2.: Example of max-pooling. The input is divided into four regions sized 2x2 pixels. Passing only the highest value of each region discards 75% of the input. Figure is taken from [45].

Pooling layers can also perform other functions, such as average pooling or L2-norm pooling but these are rarely used compared to max-pooling, which has shown to work better in practice.

Fully Connected Layer

The final layer(s) of common CNNs, such as AlexNet, are typically fully connected layers [23]. These are the same as multilayer perceptrons and are connected to all activations (i.e. detected features) in the previous layer. They combine them to a meaningful output. Due to a large number of connections, fully connected layers require a sufficiently small input volume in order to be practical. This is achieved by applying several pooling operations in the preceding layers.

4.1.2. Training

A neural network is trained by selecting all filter parameters in the convolutional- and the weights in the fully connected layer so that the network learns to map inputs to known target outputs. The backpropagation algorithm tries to solve this problem and is based on gradient descent as optimization strategy [17, pp. 200-203].
In the beginning, one input vector is propagated through the network that was initialized by some algorithm (often random values). The output of the network is compared with a target value that represents the desired output by using a loss function. Common loss functions are mean squared error or cross entropy. The loss is backpropagated and the loss of each hidden neuron is calculated individually. The gradients of each loss function can be obtained by applying the *rule of chain*. The weights of each connection are updated by subtracting a fraction of the gradient of the respective loss function. This fraction is determined by a factor called *learning rate*. Following that, the procedure is repeated for the next input vector.
A drawback of this algorithm is that it does not guarantee to find a global minimum [16]. With proper configuration, networks can nevertheless reach a good performance in practice.

4.2. Choice of Proper Network Type

An initial decision was made with regard to the network type. If the fluoroscopy runs are considered as a time series, recurrent neural networks would be a suitable choice as they have a temporal dynamic behavior. However, the fluoroscopy runs are acquired selectively during different stages of the intervention by the medical staff, therefore the scenery changes various times and furthermore acquisition gaps between the same scenery occur. Due to that, it is a requirement that the prediction of the network is based on one input sample exclusively and the idea of using recurrent neural networks was not further pursued.
Another approach is to use convolutional neural networks that have been shown to be a powerful technology for image related tasks. There have also been several applications on X-ray fluoroscopy runs. These focus on segmentation tasks (e.g. catheter segmentation [2]) and detection of abnormalities (e.g. tumor tracking [41]). Motion detection in X-ray has not been evaluated but decent results outside the medical field, as described in Chapter 2, imply CNNs as a promising approach.

4.3. Choice of Foundational Network Architecture

Zagoruyko and Komodakis evaluate different architectures that output a scalar value indicating similarity for image pairs [48]. There are two basic architectures described. A *two-channel-network* is a generic CNN where the two fluoroscopic frames are input into the first convolutional layer as a two-channel image (Figure 4.3 left). In contrast, a *siamese network* processes each frame individually in separate branches that share the same architecture and the same set of weights. The output of the branches is later concatenated and used as input to the fully connected layer (Figure 4.3 right).

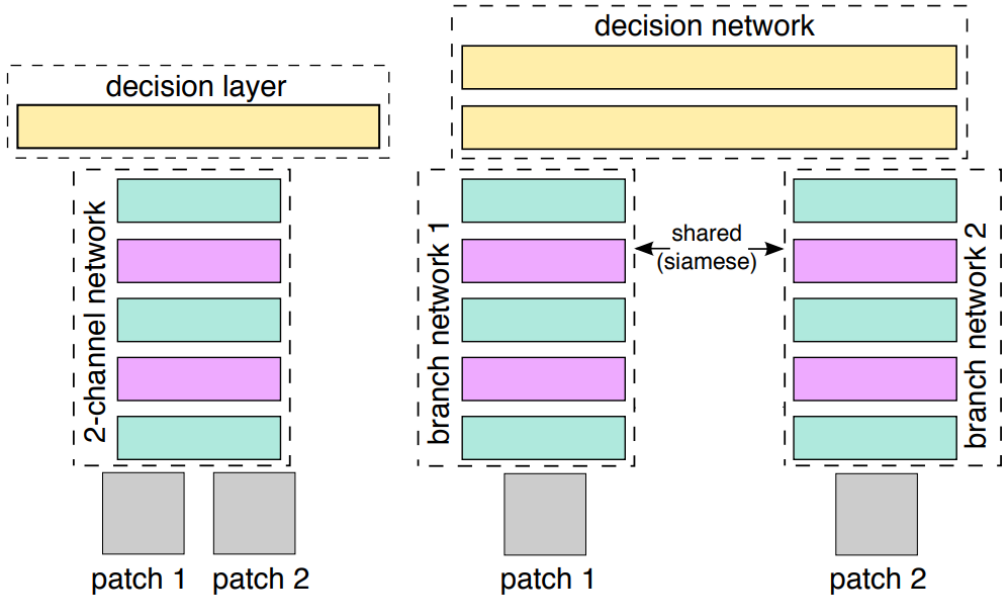


Figure 4.3.: Two-channel-network (left) and siamese network (right). Color code used: cyan = convolution, purple = max-pooling, yellow = fully connected layer. Figure is taken from [48].

According to their work, two-channel-networks are superior. Also, the FlowNet paper states that while the average performance of both network architectures is similar, the siamese network "seems to have more problems with large displacements" [13, p. 7] which might be a drawback for our application. For those reasons, we chosed to use a two-channel-network. In a later stage of the work, a siamese network was also implemented and evaluated.

4.4. Libraries and Hardware

There are numerous software tools that are designed for developing neural networks and due to computationally intensive gradient descent optimization, hardware is an important factor as well. The choices for both, soft- and hardware are described in the following.

4.4.1. Libraries

TensorFlow [1] is a math library that is mainly used for machine learning applications such as neural networks. It is written in C++ and Python and provides APIs to various programming languages. Development is mainly pursued by Google. Due to its open-source license and a large community, TensorFlow became one of the most popular tools in machine learning research. Complex calculations can be distributed across multiple machines. We used TensorFlow since it is a common choice in research and is already pre-installed on the used servers. Furthermore, numerous tutorials and an excellent documentation help to make the framework easily accessible.

Keras [10] is a high-level library that works as a wrapper for TensorFlow and other machine learning frameworks (CNTK and Theano). Keras abstracts the concepts of TensorFlow and makes it easy to build and train neural networks. The core of Keras is a model, a way to organize layer. In the simplest type of model (*Sequential()*), layers (e.g. convolutional, pooling, ...) can be stacked together and trained on a dataset. Nevertheless, Keras provides a lot of functionalities and models can be tailored for almost any use case. Due to its easy usability and the existing examples that can be adjusted to our problem, the network was built in Keras using TensorFlow as back-end.

Hyperopt [7] is a library that provides algorithms and software infrastructure for carrying out hyperparameter optimization for machine learning algorithms. It allows to set bounds for parameters and explores the search space to find the best values in a specified number of trials. Hyperopt supports several algorithms for parameters search. With Hyperas there is a wrapper to use Hyperopt with Keras. This made Hyperopt a convenient and powerful choice.

Keras Visualization Toolkit [22] is a library to visualize trained Keras models. One of the supported visualization forms are attention heatmaps that highlight the parts of the image that are focused by the network in order to generate a decision. This visualization is provided in form of a heatmap and is calculated based on an algorithm called *Grad-CAM* [38]. There are several other implementations of this algorithm available for Keras [14, 18]. This package was chosen since it is the only one that provides an example for applying the Grad-CAM algorithm to regression networks.

4.4.2. Hardware

Training happened on Nvidia graphics cards that belong to the Quadro or GeForce family and provided a much higher performance than a regular CPU.

4.5. Keras Implementation

This section first describes the project structure and all the files belonging to it. Following that, the method to validate the performance of a network is described. The remainder of the section presents the unoptimized network that was used as starting point, how it was tested with a dummy dataset and which steps were involved in optimizing it. Additionally, the effects of dataset modifications were analyzed. Finally, the performance of an implemented siamese network is presented.

4.5.1. Project Structure

Keras models were programmed in Python and the project was managed in a Git repository of the Institute of Media Informatics. New branches were created if the network architecture was significantly changed. However, most of the work was done in the *two_channel_cnn* branch, which also contains the best performing network. Figure 4.4 shows the structure with all included folders and files. The function of each part is described in this subsection.

```
breath_detection
├── network.py
├── modules
│   ├── __init.py__
│   ├── my_generator.py
│   ├── callback_lr.py
│   └── load_data_validation.py
├── metadata_training
│   ├── partition.npy
│   └── data_set.py .2 tboard
├── training_data_generation
│   ├── generate_training_metadata.py
│   ├── kick_out.py
│   └── corr.m
└── validation
    └── predict.py
```

Figure 4.4.: breath_detection repository project structure.

network.py

This is the main file that handles the training process. It loads all required modules, instantiates the generators used for passing the training data, builds the network model and finally starts the training process. All relevant modifications (hyperparameters, choice of lookup table and model architecture) are done in this file.

modules

This folder contains all the modules that are loaded within the *network.py* file.
my_generator.py contains the code that is used by the lookup table to load the training samples from the dataset and passes them to the network.
callback_lr.py contains a callback that is executed after each epoch. It saves the trained model, prints the current learning rate and evaluates the performance on the test set (the actual network only calculates the error on the validation set).
load_data_validation.py loads samples that are stored in form of a NumPy npz-array. This is the form in which the test set is stored, which requires to pass samples in the order in which they appear in a fluoroscopy run.

metadata_training

This folder contains the lookup table consisting of the two files *partition.npy* and *data_set.npy* that store the paths to the training samples and classify them into the groups "training" and "validation", as described in Chapter 3.

training_data_generation

This folder contains two scripts that are used to generate the files within the *metadata_training* folder. *generate_training_metadata.py* determines all possible samples from the fluoroscopy runs and creates the lookup table used for training.
kick_out.py takes the lookup table and allows modifications of the lookup table by setting a cap per displacement, i.e. only n randomly picked samples for each displacement are taken for training. Since samples for larger displacements are less prevalent, these can be replicated, i.e. used multiple times within training. The goal of this script is to evaluate, whether a more balanced training dataset leads to better results.
corr.m is a Matlab script that uses cross-correlation to determine the displacement of frames within a fluoroscopy run and converts this run into a folder that can be further processed to a lookup table.

validation

Although Keras calculates the error for the validation set, it is helpful to visualize a graph of the network's prediction. This allows judging, whether the error is for instance equally occurring in all frames or if certain samples have a large error while others have better results.
predict.py takes all 9 fluoroscopy runs from the test set, calculates the prediction of the network and superimposes it with the values gained from the cross-correlation.
convert2npz.py is a script that converts one fluoroscopy run into a NumPy file. This is used for the runs of the test set.

4.5.2. Validation Method

The performance of the network was validated by calculating the average mean absolute error on all samples of the validation dataset described in Section 3.6. This happens automatically after each epoch and the results are output in the console and stored in a Tensorboard file. The latter can be later used to generate a plot of the error function with respect to the epoch.
Furthermore, a callback calculates the network's performance on the test set after each epoch. This information is used to verify, whether the network truly improves or rather overfits on the validation set.

4.5.3. Unoptimized Network

The initial unoptimized network was inspired by FlowNet [13], a network used to estimate optical flow within two images (see Chapter 2 for more detail). It is the network most closely related to the problem of this thesis that was found in research. Like many networks in image related tasks, FlowNet consists of several convolutional and max-pooling layers, with the number of convolutional filters being increased in the deeper layers.
One particular difference is that FlowNet later upsamples its deep feature representation back to the resolution of an input image since the optical flow for each location of the input should be illustrated. This part is replaced with a fully connected layer and one output neuron that indicates the displacement of the diaphragm.
A FlowNet-like network was implemented by adopting a CNN example on the Keras Github page [11]. Some configuration parameters were left as they were in the example:

- Activation function: ReLu
- Optimizer: Adadelta

- Loss function: Mean Squared Error
- Kernel sizes of 5x5 and 3x3
- Neurons in fully connected layer: 128
- Dropout of 25% before each fully connected layer

The number of convolutional- and pooling layer was determined experimentally. There are two factors that are in conflict:

1. Pooling layers cause loss of spatial information and therefore should not be used too often
2. Dimensions of the feature maps need to be reduced to be computationally feasible

Using five convolutional- and pooling layers resulted in approximately three million trainable parameters. On the used hardware it took approximately one hour to train one epoch on the whole dataset. With less pooling layers, the number of trainable parameters was too high for the available dataset–even with longer training times, the network did not learn but output constant values.
Figure 4.5 illustrates the initially used network.

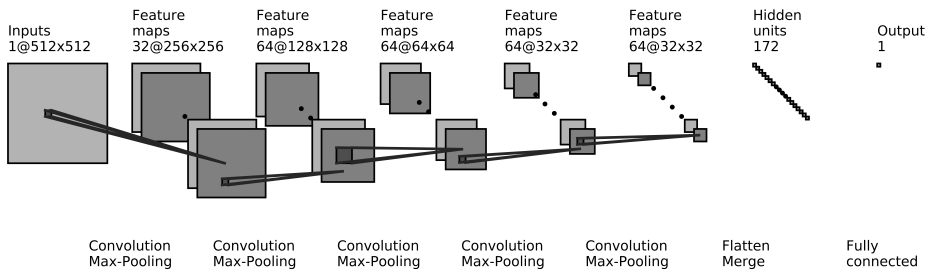


Figure 4.5.: Unoptimized convolutional neural network. It takes a two-channel-image as input and processes it in several convolutional- and pooling layers before a prediction is given by the fully connected layer.

The network was trained on the dataset for 10 epochs, which resulted in a computation time of around 10 hours. The performance of the network was measured on the validation dataset described in Section 3.6 and is plotted in Figure 4.6.
The network was able to reduce its error on the validation set over the course of training. Training effect slowed down after 5 epochs and further training beyond this point did not improve the performance significantly. As a result, the validation error did not drop below 15 pixels.

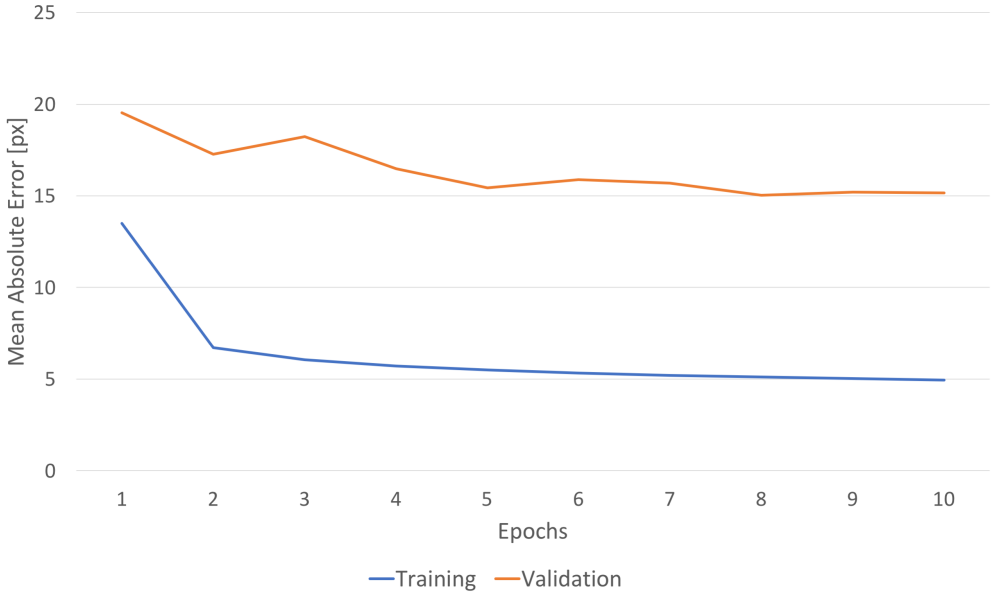
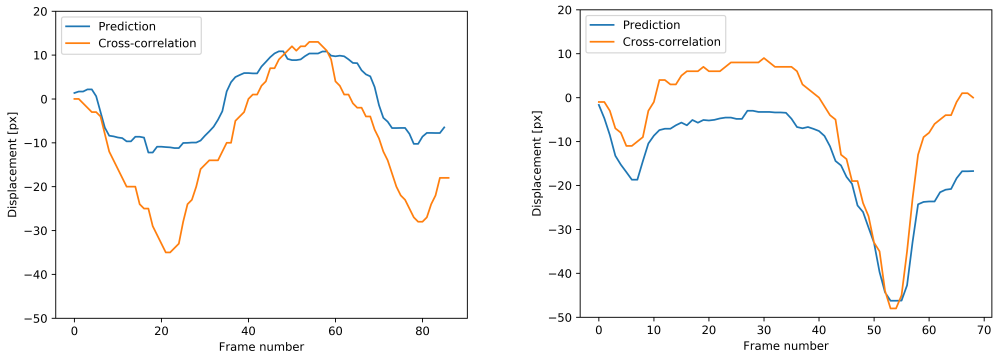


Figure 4.6.: Training- and validation error of the unoptimized network with respect to the number of training epochs. The standard deviation for the last validation run is 12.91 px, which indicates a high spread of data points around the mean.

A more detailed view on the weaknesses of the network can be gained by visualizing the predictions on fluoroscopy runs individually, instead of regarding the mean absolute error only. Exemplary, two of them are shown in Figure 4.7.



(a) Mean absolute error: 8.9 px

(b) Mean absolute error: 8.6px

Figure 4.7.: Visualization of the network's performance on two fluoroscopy runs (a, b) of the test set after 10 epochs of training. The blue curve corresponds to the prediction, the orange curve originates from tracking the diaphragm by using cross-correlation.

Even though the network's prediction roughly follows the curve of the cross-correlation, large deviations exist at particularly large absolute amplitudes of the test set, similar to Figure 4.7 (a).

4.5.4. Dummy Dataset

The unoptimized network has a high validation error and shows a weak performance in predicting large values. To make sure that it is a suitable starting point to detect the displacement of the diaphragm y-direction, a very simple but similar problem was created. The goal was to find out, whether the network is capable to predict large displacements and to what extent the dropout regularization [39] affects the performance.

The problem consists of a white square that moves within a black image as shown in Figure 4.8. Images have the same size as the fluoroscopic frames. The network is trained to detect the displacement of the white square in the y-direction.

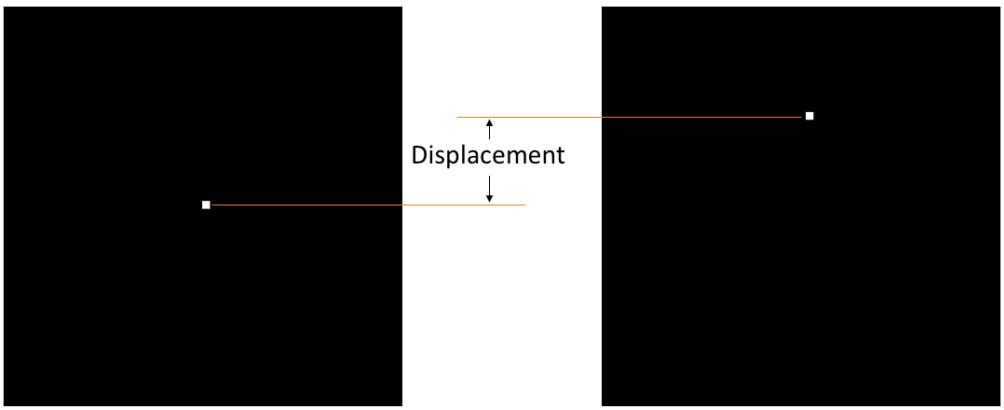


Figure 4.8.: Dummy problem to evaluate the network's capabilities. The network is trained to detect the pixel displacement of the white square with respect to a reference frame.

The training data were created from 244 frames in which the y-displacement of the white square ranges between ± 120 pixel. The displacement with respect to the frame number is illustrated in Figure 4.9. The location of the white square in x-direction is varied randomly in each frame within a range of ± 20 pixel from the center of the x-direction. Since all possible combinations of frames are generated, the dataset contains 58.322 image pairs.

If regularization is applied, dropout values are 0.25 before and 0.5 after the fully connected layer.

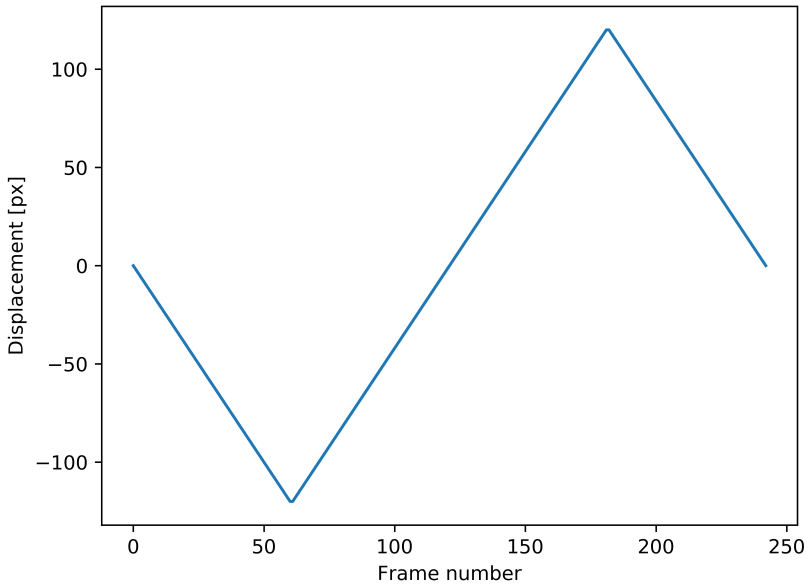
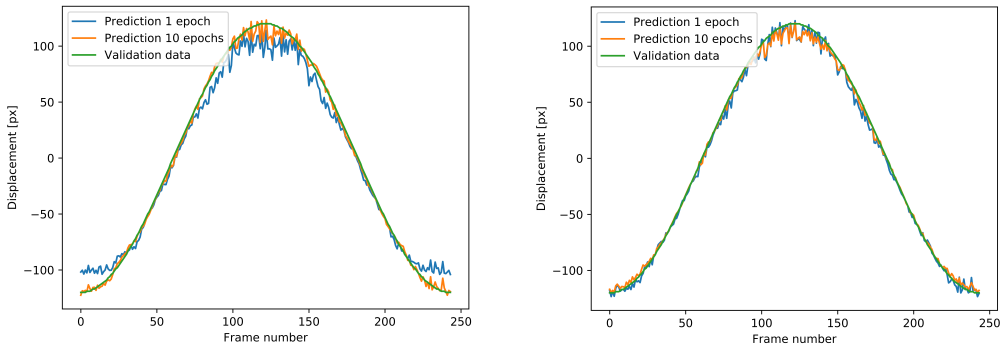


Figure 4.9.: Displacement in y-direction of the white square within the training data. A random shift in x-direction is added in order to make the dummy problem more realistic.

Figure 4.10 shows the results of the dummy network with (a) and without (b) applied dropout regularization.



(a) Dropout applied

(b) No Dropout

Figure 4.10.: Visualization of the network's performance on the validation dataset after being trained for one (blue) and 10 (orange) epochs. The actual curve of the validation data is additionally drawn green. Evaluation was done for a network with dropout applied (a) and without a dropout layer (b).

The mean validation error is listed in Table 4.1.

Network	Training Epochs	Mean Error [px]
Dropout	1	5.64
	10	2.03
No Dropout	1	2.79
	10	2.21

Table 4.1.: Mean error of the network with respect to the dummy validation dataset.

The dummy dataset showed that:

- The network is able to solve a very simplified version of the diaphragm tracking with reasonable accuracy
- The network does not require extensive training but rather shows good results after one epoch of training on a dummy dataset that is 16 times smaller than the dataset containing the fluoroscopy runs.
- Dropout slows down convergence, however, with enough training, it does not cause the performance of the network to be worse
- Accuracy tends to be worse and noisy in the range of large absolute displacements

4.5.5. Hyperparameter Optimization

Besides the weights and filter parameters that are optimized by the training algorithm, there are numerous hyperparameters that influence the performance of the network and need to be specified before training. These hyperparameters include:

1. Number of layers
2. Number of filters
3. Activation function
4. Optimizer configuration
5. Loss function
6. Batch size
7. Training duration
8. Dropout values

The number of possible parameters and configurations is too large in order to automate the search. In the beginning, each parameter was regarded once at a time, and it was investigated, whether a change in one or the other direction brought significant improvements. For instance, we added one convolutional layer to the unoptimized network which caused the mean validation error to increase by 5.2%. If instead one convolutional layer was removed, the mean validation error increased by 36%. Therefore, the number of layers were left as they are and it was continued to apply this procedure to the number of filters and most other hyperparameters.

Some numerical parameters (dropout, filter size and number of filters in each layer) were optimized automatically by letting Hyperopt find good values in a specified number of trials. These parameters have a large range of reasonable values, therefore it was too cumbersome to optimize them manually. For the filter sizes, combinations with values of 3x3, 5x5, 7x7 and 9x9 were tried, while the dropout rate can be any value between zero and one.

As result, most of the hyperparameters chosen in the unoptimized network were already properly chosen. Adjusting most of them in one or the other direction caused the network either not to work anymore (constant output of mean target value of dataset), worsen the performance or had neglectable effects. Filter sizes and the number of filters had almost no influence on the validation error. Regularization, i.e. dropout values, and the number of neurons in the fully-connected layer were the only values that lead to a significant improvement ($\sim 20\%$) of the validation performance.

4.5.6. Dataset Modification

In order to evaluate its influence, the amount of data was increased twice during the course of the work. This can affect the network for two reasons:

1. The diversity of the data used for training is increased, i.e. while keeping the amount of training data constant, it is composed of more patients/fluroscopy runs

2. The amount of data used for training is increased, i.e. the number of patients/fluroscopy runs is kept constant and the amount of training data is increased

In a first test, it was evaluated, how more diverse data influence the performance. This was done by keeping the number of samples per displacement constant. If the pool of data from which these samples are randomly picked is increased, the diversity increases, while the amount of data used for training stays the same. The results after 10 epochs of training are listed in Table 4.2.

Test #	Number of Fluoroscopy Runs	Data Increase	Mean Validation Error [px]	Improvement
1	104	-	10.2	-
2	120	24%	9.5	7.3%
3	150	42%	9.5	7.3%

Table 4.2.: Results of a network trained with data that consists of fluoroscopy runs from a different number of patients. The column "Data Increase" lists how much the dataset was increased compared to the dataset in Test 1 and the column "Improvement" lists the percentage error reduction compared with Test 1.

The first increase in training data diversity caused an improvement by 7.3%. This trend is not confirmed for another increase but there is no negative effect in form of a higher validation error.

In a second test, the influence of more training data was evaluated for a constant diversity (i.e. the number of fluoroscopy runs from which the data comes from) is kept constant. This was done by setting different caps for the number of samples per displacement that were used for training. If the cap is set higher, the training data becomes more imbalanced since absolute high displacements occur not often enough to provide enough samples to reach the cap–small displacements are overproportional represented. The number of fluoroscopy runs is 150 and the mean validation error was calculated after 10 epochs of training. The network has a dropout regularization of 0.5 before and after the first fully connected layer. Results are listed in Table 4.3.

Test #	Max. number of samples per displacement	Mean Validation Error [px]	Improvement
1	2.500	10	-
2	5.000	9.2	8.7%
3	7.500	8.0	25%
4	10.000	7.8	28.2%
5	all	6.8	47.1%

Table 4.3.: Results of a network trained with data capped at a different number of samples per displacement. The "Improvement" column refers to the validation error reduction compared with Test 1.

Increasing the size of the training data significantly reduces the validation error. Concerns that larger, but more imbalanced training data worsens the overall performance were not confirmed. If the impact of the dropout layer is weakened, imbalanced training data might become relevant again. However, in this configuration, the best result was achieved while using all available training data.

4.5.7. Siamese Network

The optimized network was changed in order to form a siamese network. Both images of a sample are processed in an identical stream consisting of five convolutional- and polling layers. The parameters are the same as in the two-channel-network, except the depth of the last convolutional layer was reduced from 128 to 64. This was done so that after concatenating both streams, the depth of the resulting input to the fully connected layer is 128 again. By that, the number of trainable parameters of approximately 4,000,000 is almost identical with the two-channel-network and their complexity is similar. The network structure is shown in Figure 4.11.

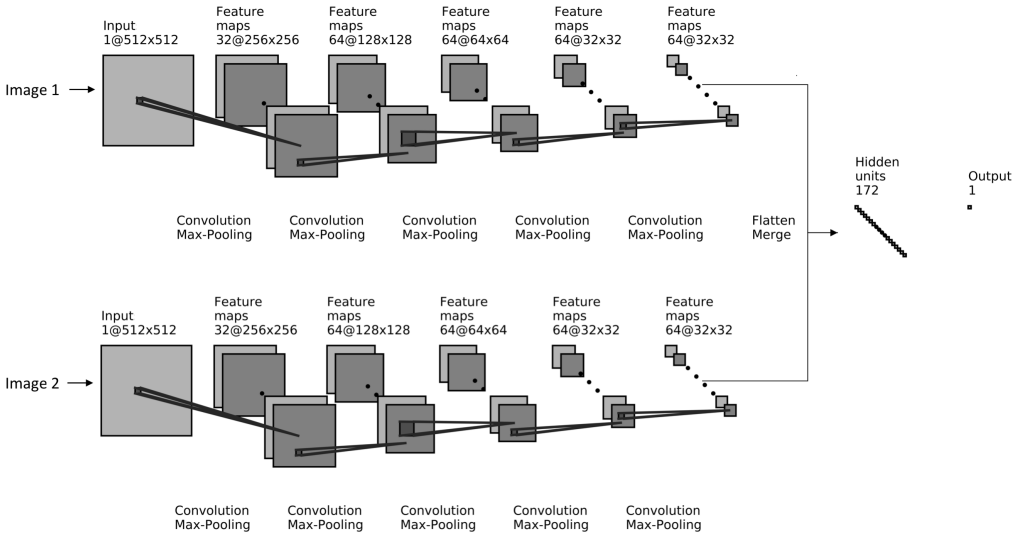


Figure 4.11.: Siamese network. Both input images are processed in separate but identical streams. Their results are merged after the last convolutional-/pooling layer.

The evaluated siamese network was trained for 10 epochs on the full dataset. The results are shown in Figure 4.12

The network was trained without dropout regularization since applying it causes the network to have a constant output of zero. Due to this, the training error can drop to a value of about one pixel. The two image streams are concatenated before the fully connected layer and randomly "dropping" units from one or the other stream intuitively also does not make sense. Using a deeper fully connected layer with dropout did not yield a better result.

The mean absolute validation error increases over the epochs, which is a sign that the network is not able to generalize the problem. Overall the siamese network struggles with the same validation fluoroscopy runs than the two-channel-network. Due to time

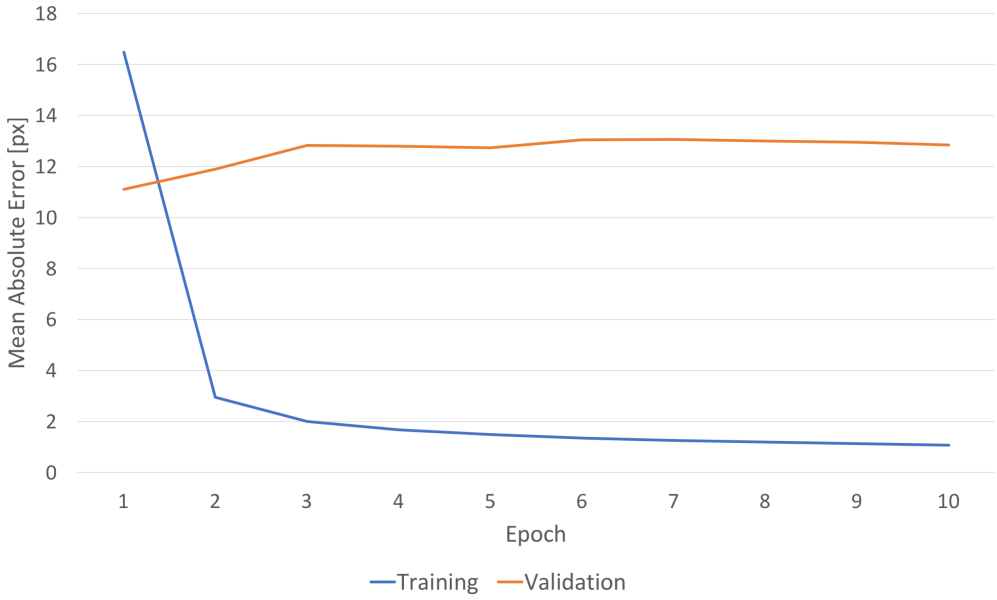


Figure 4.12.: Mean absolute error of the final network for training set (blue) and validation set (orange) for 10 epochs of training.

constraints, the siamese network configuration was not optimized as it was done with the two-channel network. Since the average error is significantly higher, it is unlikely that optimizing network will lead to a comparable or better performance.

5. Evaluation

In the beginning of this chapter, the results from the optimized network are described. This includes analysis of the error on the validation dataset and plots of the prediction of selected fluoroscopy runs from the test set. The results can be directly compared with the performance of the unoptimized network in Section 4.5.3 in order to assess the improvement.
Following that, heatmap plots that highlight regions of the input that are important for the network's decision are presented, which gives an insight into the functionality of the network.
Afterwards, the performance of the network on fluoroscopy runs that do not contain the diaphragm is analyzed. The fluoroscopy runs used in these cases cannot be labeled, since the cross-correlation algorithm fails. For this reason, the quality of the prediction is judged by visually comparing fluoroscopy run and prediction.
The section concludes with a critical discussion of the used methods.

5.1. Training Results with Final Dataset

Results of the network after being trained for 10 epochs are shown in Figure 5.1.

Compared to the unoptimized network, as presented in Section 4.5.3, the validation error is more than halved. Nevertheless, with a mean error of about seven pixels, there are still significant deviations occurring. The training error decays rapidly for two epochs, after that there is only a minor improvement visible. This is mostly due to the dropout regularization layer. If removed, the training error falls well below one pixel, but the validation error increases and by that shows the importance of regularization.

An improvement of the prediction quality is obvious when visualizing two examples from the test set again. The predictions of the optimized network (see Figure 5.2) are better than the predictions of the unoptimized network (see Figure 4.6). Within the test set, the error is in average 4.15 px with a standard deviation of 3.83 px and varies across the fluoroscopy run in a range between 2.2 px and 9.1 px. Large errors mainly originate in an underestimation of large displacements.

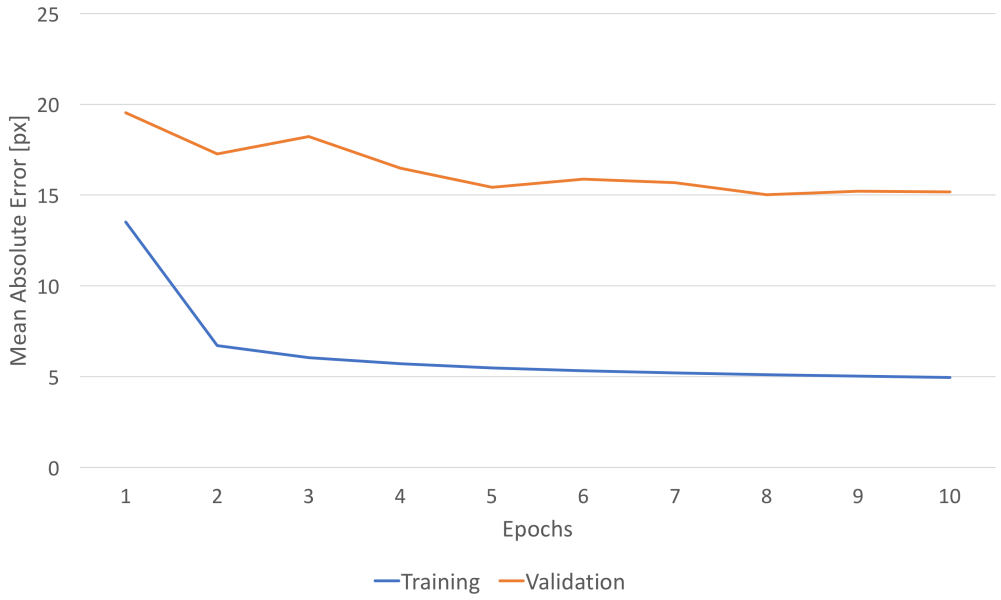
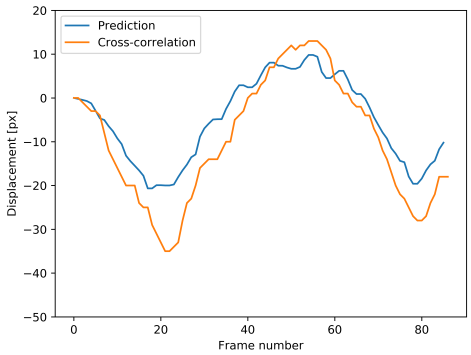
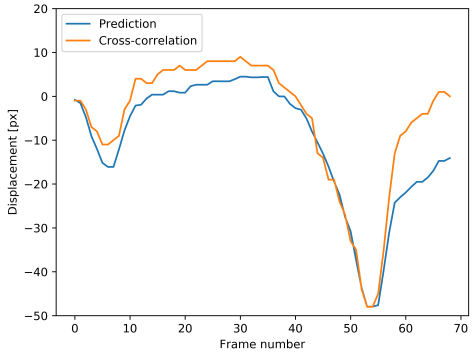


Figure 5.1.: Mean absolute error of the final network for training set (blue) and validation set (orange) for 10 epochs of training. The standard deviation for the last validation run is 8.95 px, which indicates high spread of data points around the mean.



(a) Mean absolute error: 5.1px



(b) Mean absolute error: 5.0px

Figure 5.2.: Visualization of the network's performance on two randomly picked fluoroscopy runs (a, b) of the test set after 10 epochs of training. The blue curve corresponds to the prediction, the orange curve corresponds to tracking the diaphragm by using the cross-correlation.

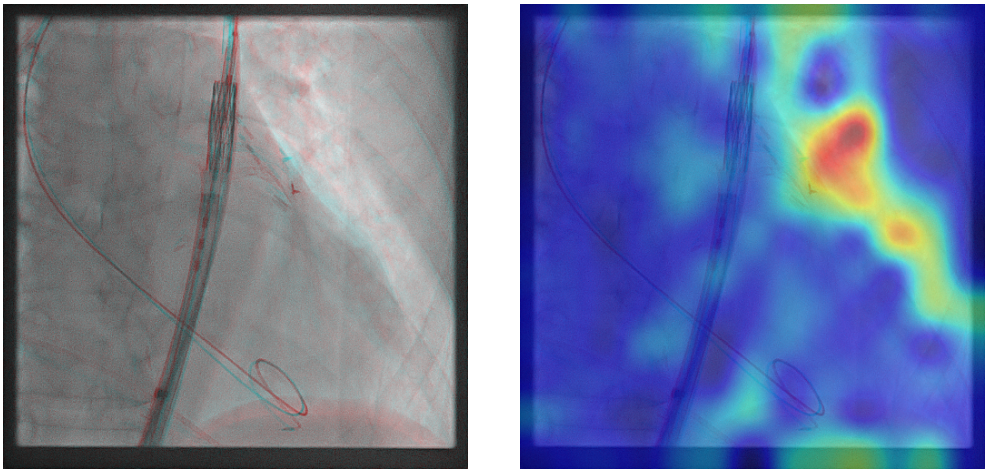
5.2. Heatmap Plots

Heatmap plots were created for fluoroscopy runs from the test set with the Keras Visualization Toolkit. These plots highlight how many different regions of the image contribute to the network's decision. The heatmaps are a modification of the class activation maps generated by the Grad-CAM algorithm [38] to make it applicable to regression problems. In classification tasks, Grad-CAM takes the final convolutional layer and weights every channel within that layer with the gradient of a certain class with respect to the channel. As a result, it shows how intensely the input activates the different channels and how important each channel is with regard to the class.
Basically, the algorithm calculates what parts of the input increases the output for a specific class. For the regression network we use, it is more interesting to see, what parts of the input cause the output not only to increase but also to decrease or maintain the current predicted value. Therefore, we also need to consider small and negative gradients. The modifications are mostly taken from an example of a network that predicts steering wheel positions [34]. For analyzing negative gradients we simply negate all gradients and then we can pass them to the Grad-CAM algorithm. If we want to find parts of the input that have little influence on the predicted value, we focus on small gradients by passing the reciprocal of the gradients to the algorithm. Therefore, for each input three heatmaps are calculated which illustrate:

1. Regions of the input that cause the prediction to become large, i.e. positive
2. Regions of the input that cause the prediction to become small, i.e. negative
3. Regions of the input that have no substantial effect on the prediction

In this section, all heatmaps highlight the parts that contribute to a negative value. However, similar results are drawn if the heatmap plots causing the prediction to become positive are analyzed.
Since the training data of the network was generated by tracking the diaphragm, intuitively one would expect that the network also primarily pays attention to this structure. This is not necessarily the case, as the heatmap plots reveal.
One example is shown in Figure 5.3 that highlights regions that contribute to a small (negative) prediction. The focus of the network is spread across several parts of the fluoroscopic frame and the diaphragm seems to be not particularly important.
Furthermore, attention regions of the network are not constant but rather jump to different parts of the fluoroscopic frame. It seems that the network is drawing its prediction from a lot of different features, and even if the diaphragm seems to play a minor role here, the result can be similar to the cross-correlation results that were calculated based on the diaphragm. The high accuracy is obvious in the corresponding overlay of

cross-correlation and the network's prediction in Figure 5.3.



(a) Sample (b) Sample with heatmap overlay

Figure 5.3.: Based on the sample (a), a heatmap overlay is created and visualized (b). According to the heatmap, the network pays only minor attention to the diaphragm (bottom right of frames). The focus of the network is mainly on the heart contour.

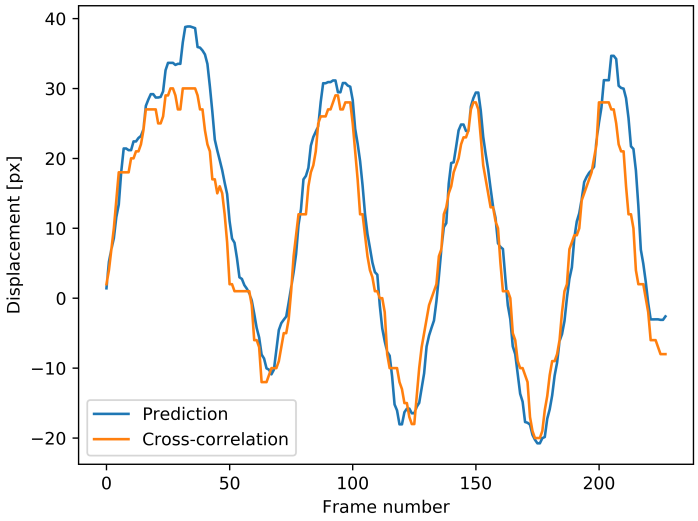
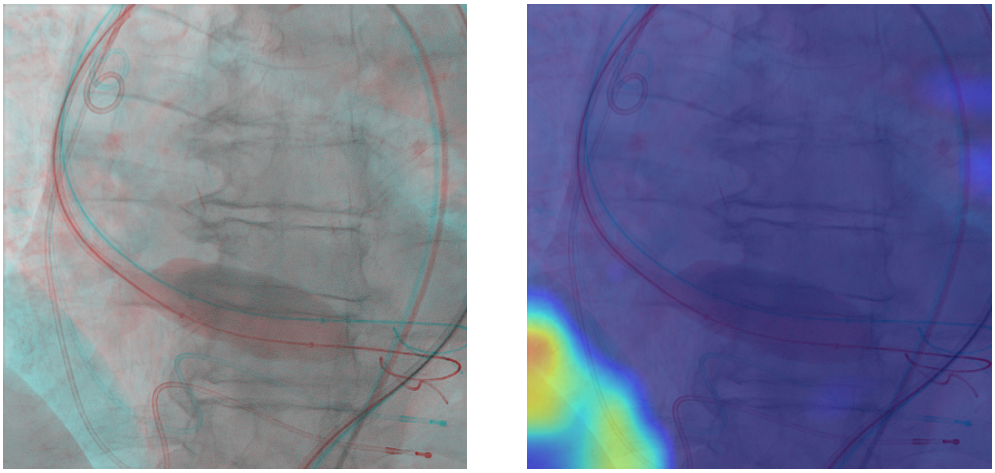


Figure 5.4.: Comparison of the network's prediction (blue) with the result from cross-correlation (orange). Calculation was done on the fluoroscopy run of the sample in Figure 5.3.

In contrast, the network focuses its attention almost exclusively on the diaphragm in some fluoroscopy runs (e.g. Figure 5.5). Nevertheless, the prediction has severe problems with detecting large displacements as can be seen in Figure 5.6.



(a) Sample (b) Sample with heatmap overlay

Figure 5.5.: Based on fluoroscopic frame pairs (a), a heatmap overlay is created and visualized (b). The heatmap shows the strong focus the network puts on the diaphragm.

The conclusion we derived by analyzing the heatmaps is that the network does not necessarily base its decision on the diaphragm. This is interesting since the network was trained on the displacement of the diaphragm and tracking it seems to be comparably simple from a human perspective. However, the focus of the network varies from fluoroscopy run to run and even within a run the parts that contribute to the prediction can be different from sample to sample. It is noticeable that the network's attention is mostly focused on prominent shapes (e.g. catheter). Respiratory motion often can be observed at the lung vessels and bronchioles. However, these fine structures have a low contrast and are prone to deformation. The network seems to be unable to extract this information.

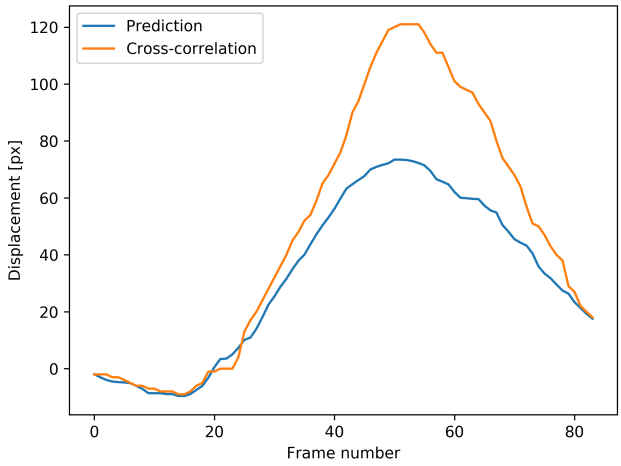


Figure 5.6.: Comparison of the network's prediction (blue) with the result from cross-correlation (orange). The calculation is based on the fluoroscopy run of the same sample shown in Figure 5.5.

5.3. Data without Diaphragm

It is difficult to track respiratory motion with cross-correlation if no diaphragm is present in the fluoroscopy run. To analyze, whether the neural network can handle these situations better, 12 runs from different patients were picked from the clinical system as test cases. These have in common that the diaphragm is either not visible or disappearing in the course of the run. The optimized network was used to predict displacements. In 9 of 12 runs, the network's predicted waveform has a clear correlation with respiratory motion in the run. This confirms the results from the heatmap plots (Section 5.2) that the diaphragm is not the network's sole region of interest.

In the three failing predictions the motion can be clearly seen at lung vessels and bronchioles but as discussed in the previous section, the network cannot detect these structures. One example is shown in Figure 5.7. Heartbeats are strongly visible, but their effect can be reduced by fitting a polynomial curve into the data (orange line in Figure 5.7). This gives the waveform of the respiratory motion (expiration-inspiration-expiration) but the values seem too small compared to what can be seen in the fluoroscopy run.

A better result is shown in Figure 5.8. Although the diaphragm disappears from the fluoroscopy run if the displacement is smaller than -30 pixels, the network is still able to detect a much higher displacement in that direction. If visually compared to the fluoroscopy runs, this seems to be a plausible result.

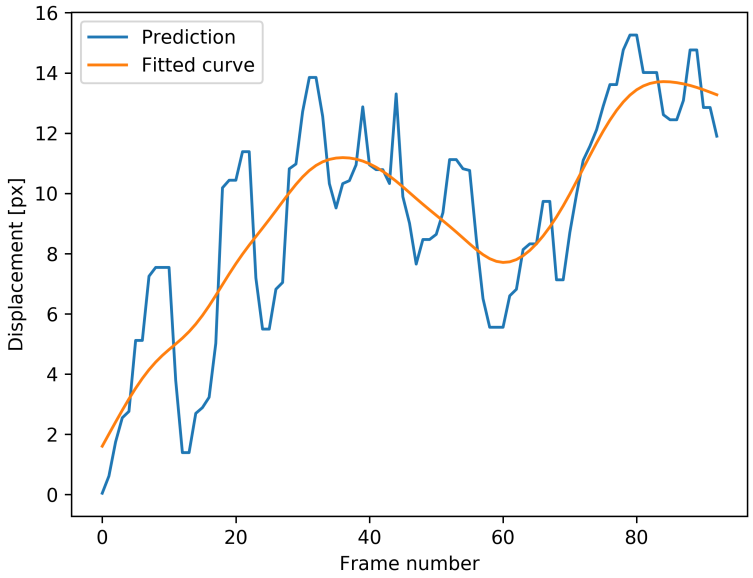


Figure 5.7.: Motion prediction of fluoroscopy runs without the diaphragm. The waveform of respiratory motion can be obtained by fitting a polynomial function (blue) that suppresses the cardiac components of higher frequency.

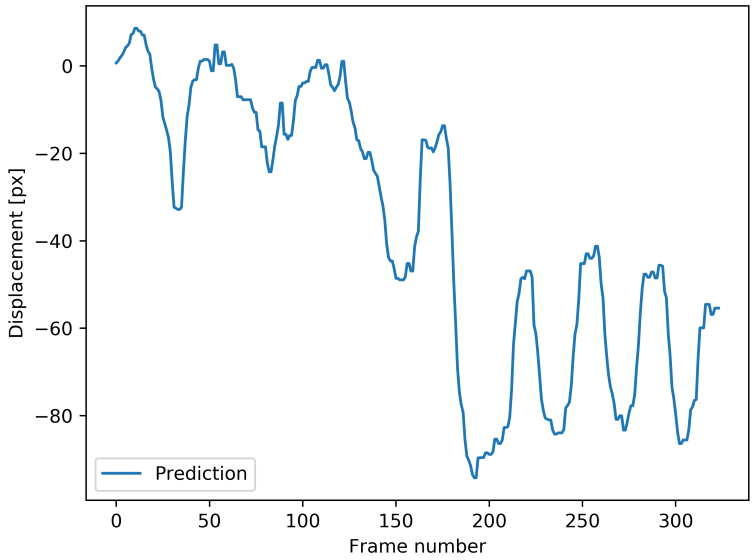


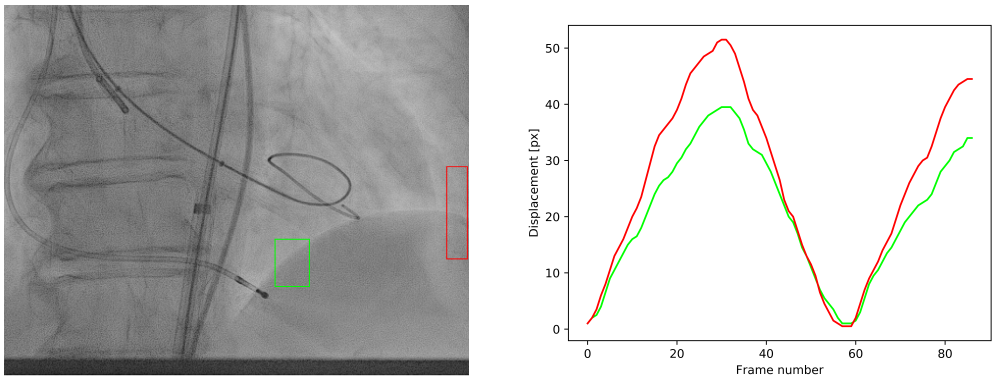
Figure 5.8.: Motion prediction of fluoroscopy runs. The diaphragm is not visible at a displacement of below -30 pixels.

Noticeable is that for both examples the predicted curves seem to correspond strongly to the motion of the catheter, which is a particular ROI according to the heatmaps. If

the fluoroscopy runs with failing predictions are visually analyzed with respect to the catheter's motion, the results seem to be plausible. In these cases respiratory motion has little influence on the catheter while heartbeats are clearly visible.
This observation could not be made in the validation data containing the diaphragm. It was not possible to perceive a clear correlation between the motion of the catheter and the quality of the prediction. Even if the catheter is stationary or strongly affected by comparably high cardiac motion, accuracy can be very high. In contrast, an approximately congruent motion of the catheter and the diaphragm does not necessarily lead to a good prediction.

5.4. Drawbacks of Dataset Quality

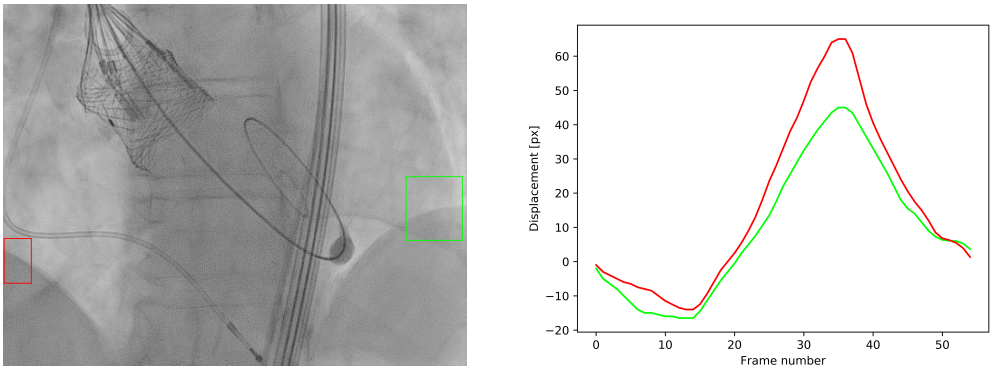
The network was trained on the displacement of the diaphragm and for determining target values we used the simplified assumption that the displacement of the whole diaphragm can be expressed by a single scalar value. However, analysis of the dataset has shown that the diaphragm is very prone to deformation. The target values can be very different, depending on the ROI that is used for calculating the cross-correlation. One example is illustrated in Figure 5.9. The left image shows an excerpt from a fluoroscopic frame with two ROI highlighted. The result of the cross-correlation calculated on these ROIs has a mean absolute deviation of 5.6 px and a maximum of 12.8 px.



(a) Different ROIs (b) Results of cross-correlation

Figure 5.9.: Cross-correlation results can vary greatly depending on the ROI. The two selected ROIs in (a) result in curves (b) that have a mean absolute deviation of 5.6 pixels and a maximum absolute deviation of 12.8 pixels with respect to each other.

The diaphragm can be visible on both sides – right and left – of a fluoroscopy run. In these cases, the diaphragm motion can differ strongly between sides (see Figure 5.10).



(a) Different ROIs (b) Results of cross-correlation

Figure 5.10.: Cross-correlation results can vary particularly strong if the diaphragm is present on both sides of the image. The two selected ROIs in (a) result in curves (b) that have a mean absolute deviation of 8.1 pixels and a maximum absolute deviation of 22.5 pixels.

During training data generation it was underestimated that deviations can be that large depending on the selected ROI. Therefore, the ROI was applied to any location on the diaphragm and if the waveform appeared plausible when being visually compared with the fluoroscopy runs, it was added to the training data. This means, the target values are fuzzy and a large range of values can be considered 'correct'. In retrospect, it might had been a good idea to restrict potential ROIs to a consistent part of the diaphragm.
This was tested for an abstract that was submitted to the CARS congress 2019. Here all fluoroscopy runs in which the ROI is not in the most right 17% of the frame were excluded. Due to this, the training dataset size shrunk by a third. Unfortunately, also validation and test set were altered and therefore results of other runs on the full dataset could not be directly compared. If we only compared the performance on fluoroscopy runs that exist in the full and reduced dataset, the network that was trained on data with restricted ROI showed a by 10% lower validation error.
Comparing the displacements determined by cross-correlation and network prediction often show a large deviation in the range of large absolute displacements. This is also the range in which target values can differ strongly depending on the selected ROI.
Due to this, it is difficult to evaluate if that error is a weakness of the network itself or of the dataset. The dummy dataset in Section 4.5.4 showed indeed that error rates in large displacements are higher compared to small ones. Still, the prediction for the dummy dataset has a much higher accuracy compared to the dataset containing the fluoroscopy runs.

The non-uniform motion of the diaphragm might hinder the network to identify the diaphragm as the sole object to track. Since only punctual parts of the diaphragm's motion correspond to the target values, the network also identifies these in other moving parts, e.g. catheter or heart contours. This would explain the large variations within the heatmaps.

Variation of the absolute error within validation and test set is very high. Fluoroscopy runs were analyzed visually but it was not possible to find a property that correlates with a small or large error. One idea was that strong non-uniform motion of the diaphragm correlates with a worse prediction. This was difficult to evaluate visually, therefore, for each fluoroscopy run in the validation set the non-uniformness of the diaphragm's motion was quantified.

Unfortunately, calculating displacements with cross-correlation in Matlab often fails, for instance, due to an interfering catheter. In most cases, it was not possible to find two significantly different ROIs that lead to plausible results which can be compared, like in Figure 5.9 and 5.10.

However, for a small number of samples, this evaluation can be done manually. We regarded each validation fluoroscopy run individually and selected the frame pair in which the diaphragms in both frames have the highest absolute displacement with respect to each other. Then we analyzed, to what extent this displacement varies if we measure it at different locations on the diaphragm. Our approach for doing this was to create an overlay of both frames with 50% transparency in the upper layer. In this overlay we can see the location of the diaphragm of both frames and can determine the displacement simply by using an image viewer with ruler feature (here Krita [20]). We determined the displacement on the farthest visible location left and right of the diaphragm and calculated the difference we observed here.

Table 5.1 lists the results.

Validation Run No.	Absolute Deviation [px]	Relative Deviation	Mean Validation Error [px]
1	2	7.7%	4.15
2	9	24%	5.11
3	5	14.3%	3.87
4	15	30%	7.57
5	5	11.1%	8.61
6	1	2.3%	4.76
7	7	17.5%	2.62
8	20	30.7%	2.89
9	20	14.3%	17.7
10	6	20%	3.46
Average	9.0	17.2%	6.07

Table 5.1.: Results of analyzing the frame pair with maximum displacement in each validation fluoroscopy run. The column "Absolute Deviation [px]" lists the deviation when measuring the displacement of the diaphragm on its left and right side and calculating the absolute difference of both. The column "Relative Deviation" shows the percentage this deviation forms with respect to the maximum displacement. There is no obvious correlation between deviation and mean validation error. Note that the average mean validation error is smaller than in Figure 5.1 since all runs are weighted equally, regardless of the duration of them.

With a correlation coefficient of 0.49 there is no strong correlation between the non-uniform motion of the diaphragm and the mean validation error. For instance, validation run no. 8 shows the highest absolute and relative deviation but in contrast, ranks second with respect to the mean validation error. In conclusion, it remains unknown what properties of a fluoroscopy run can be used to estimate the quality of the prediction.

5.5. Interpretation

The network has been able to approximate the functionality of the cross-correlation to a certain extent. While it reliably detects phase transitions in the respiratory cycle, the displacements can significantly differ from the target values–especially at large absolute displacements. For the dummy dataset this observation could not be made to this extent. These results suggest that the high error values in predicting large displacements are not an issue of the network itself but of the higher complexity of the dataset containing the fluoroscopy runs.

During this work, the error could be reduced by adjusting the network hyperparameters and by increasing the dataset size. It is likely that a further increase of the dataset size, especially with fluoroscopy runs from new patients, leads to higher accuracy.

In comparison to dataset modification, the hyperparameters and network architecture seemed to be of minor importance. This is an indication that the dataset is not large enough to let the network generalize upon the problem. The heatmap plots revealed that the networks attention is varying strongly across different fluoroscopy runs and presumably modifications of the dataset might also change the focus of the network's attention and therefore lead to other results.

As expected from literature, a siamese network lead to inferior results. Although hyperparameter tuning was not done to the same extent as in the two-channel-network, it is unlikely that the overall result would have drastically changed.

To summarize, it can be stated that the network shows promising results. With further improvements the accuracy will likely increase and specially the ability to generate predictions on fluoroscopy runs without diaphragm and the fact that a manual ROI selection is not required with this method are advantages compared to applying the cross-correlation. However, it still has to be analyzed, how accurate the prediction is exactly in these cases.

Due to the large variations in target values depending on the selected ROI, it is not possible to finally assess the quality of the network. This would require training data with more consistent target values. At the same time, this indicates that tracking the diaphragm for motion compensation, as is currently done, is unreliable.

6. Conclusion

The main contribution of this thesis was to implement a convolutional neural network as a novel approach to extract respiratory motion within fluoroscopic frames. This motion was approximated by using the displacement of the diaphragm in the head-feet direction. An advantage of the diaphragm is that it is relatively unaffected by cardiac motion and its prominent shape allows being tracked in a semi-automatic process. The required dataset was created by selecting a ROI covering parts of the diaphragm and tracking it throughout the consecutive frames in the fluoroscopy run. A total of 150 runs were processed and used in either the training-, validation- or test set. Implementation of the network was done in Keras using TensorFlow as back-end. The unoptimized network was based on an architecture used for optical flow prediction. From this starting point, the performance could be gradually optimized through dataset modifications and hyperparameter tuning. Finally, the network was able to reliably detect the respiratory waveform and therefore detect the respiratory rate and phase. The quality of the prediction varies between different fluoroscopy runs of the validation set. Especially large absolute displacements are often underestimated by the network, i.e. runs with large displacements have usually a higher mean absolute error. It was not possible to identify other properties that correlate with the quality of the prediction.

Heatmap plots revealed that although being trained on the diaphragm, the network does not necessarily have a strong focus on this shape. Predictions are sometimes rather based on a composition of numerous different objects such as the diaphragm, hear contours or the catheter. In other cases, the diaphragm is the sole object of interest for the network. Nevertheless, there was no obvious correlation between the network's focus and the quality of the prediction.

With respect to the research questions, it can be stated that convolutional neural networks are to some extent capable to detect displacements of the diaphragm. Results of cross-correlation and neural network are similar for many fluoroscopy runs but significant deviations in runs with a large range of displacements occur. For frame pairs that do not contain the diaphragm, cross-correlation is not applicable. In contrast, the neural network is often still able to provide a prediction that appears to correlate with respiration. In these cases, the prediction seems to be mainly based on the motion of the catheter if the results are visually compared with the fluoroscopy run. The hope that

a neural network might be able to also track the motion of bronchioles and lung vessels that appear with a very low contrast did not fulfill.

During the work, it became obvious that choosing the diaphragm as the shape to determine respiratory displacements has severe drawbacks. Deformation causes the motion of the diaphragm to be non-uniform. Describing its motion with a single scalar value, as it was done in this thesis, is not very accurate. As the evaluation has shown, the displacement of the diaphragm can have location-dependent variations of more than 30%. Target values can, therefore, be rather different for the same diaphragm. This limits the validity of using the mean absolute error as measure for the prediction quality. Although this work has shown that convolutional neural networks are capable to extract respiratory motion from fluoroscopic frames, several aspects are advisable for future work:

- *Dataset type:* Target values should be obtained from shapes that are less prone to deformation. Ideally, it is possible to label the displacement of the catheter. Since tracking algorithms usually fail on tracking the catheter, it might be necessary to do this manually. If the target values are generated from a shape with a uniform motion, it would be interesting, whether the network learns to base its prediction only on its motion instead of composing motions from multiple parts of the input.

- *Dataset size:* The accuracy of the prediction varies strongly across the validation fluoroscopy runs. It is advisable to increase the size of the validation set in order to reduce the risk of overfitting the network to the validation set during hyperparameter tuning.

A. Appendix

A.1. Program Code

The appendix includes the code for the network implementation and the data generator module as key parts of the implementation. All scripts can be found in the corresponding Git repository.

A.1.1. Network Implementation

```
'''Trains a simple convnet on a custom dataset'''

from __future__ import print_function
import keras
from keras.models import Sequential, load_model
from keras.layers import Dense, Dropout, Flatten
from keras.layers import Conv2D, MaxPooling2D, Conv3D
from keras import backend as K
import numpy as np
from modules.my_generator import DataGenerator
from modules.callback_lr import LearningRate
from modules.load_data_validation import *

# Parameters
params = {'dim': (512,512),
          'batch_size': 32,
          'n_classes': None,
          'n_channels': 2,
          'shuffle': True}

input_shape = (512, 512, 2)

# Load Datasets
partition = np.load('./metadata_training/partition.npy').item()
data_set = np.load('./metadata_training/data_set.npy').item()
```

```
# Instantiate Generators
training_generator = DataGenerator(partition['train'], data_set, **params)
validation_generator = DataGenerator(partition['validation'], data_set, **params)

#Build model
model = Sequential()
model.add(Conv2D(32, kernel_size=(5, 5),
                 activation='relu',
                 input_shape=input_shape))
model.add(MaxPooling2D(pool_size=(2, 2)))
model.add(Conv2D(64, kernel_size=(5, 5),
                 activation='relu',
                 input_shape=input_shape))
model.add(MaxPooling2D(pool_size=(2, 2)))
model.add(Conv2D(64, kernel_size=(3, 3),
                 activation='relu',
                 input_shape=input_shape))
model.add(Conv2D(128, kernel_size=(3, 3),
                 activation='relu',
                 input_shape=input_shape))
model.add(MaxPooling2D(pool_size=(2, 2)))
model.add(Conv2D(128, (3, 3), activation='relu'))
model.add(MaxPooling2D(pool_size=(2, 2)))
model.add(Conv2D(128, (3, 3), activation='relu'))
model.add(MaxPooling2D(pool_size=(2, 2)))
model.add(Conv2D(128, (3, 3), activation='relu'))
model.add(MaxPooling2D(pool_size=(2, 2)))
model.add(Dropout(0.25))
model.add(Flatten())
model.add(Dense(128, activation='relu'))
model.add(Dropout(0.5))
model.add(Dense(1, activation='tanh'))

# Compile model and print summary
model.compile(loss='mean_squared_error',
              optimizer=keras.optimizers.Adadelta(),
              metrics=['mae'])
print(model.summary())

# Define callbacks for TensorBoard and printing the learning rate
print_lrate = LearningRate()
callbacks = keras.callbacks.TensorBoard(log_dir='./tboard',
histogram_freq=0, write_graph=True, write_images=True)
```

```
# Train model
model.fit_generator(generator=training_generator,
                    validation_data=validation_generator,
                    use_multiprocessing=True,
                    workers=4, epochs=10, callbacks=[callbacks, print_lrate],
verbose=1)

# Save model
model.save('net.h5')
```

Listing A.1: Keras network implementation

A.1.2. Generator Module

```
'''Skript to generate Dataset from folders
adapted code from https://stanford.edu/~shervine/blog/keras-how-to-generate-data-on-the-fly.html'''

import numpy as np
import keras
import cv2
import random
from datetime import datetime

class DataGenerator(keras.utils.Sequence):
    'Save passed parameters from main function'
    def __init__(self, list_IDs, data_set, batch_size=32, dim=(512,512), n_channels=1,
                 n_classes=10, shuffle=True):
        'Initialization'
        self.dim = dim
        self.batch_size = batch_size
        self.data_set = data_set
        self.list_IDs = list_IDs
        self.n_channels = n_channels
        self.n_classes = n_classes
        self.shuffle = shuffle
        self.on_epoch_end()
        random.seed(datetime.now())

    def __len__(self):
        'Denotes the number of batches per epoch, by default all samples are trained once per epoch'
        return int(np.floor(len(self.list_IDs) / self.batch_size))
```

```
    def __getitem__(self, index):
        'Generate one batch of data'
        # Generate indexes of the batch
        indexes = self.indexes[index*self.batch_size:(index+1)*self.batch_size]

        # Find list of IDs
        list_IDs_temp = [self.list_IDs[k] for k in indexes]
        # Generate data
        X, y = self.__data_generation(list_IDs_temp)
        return X, y

    def on_epoch_end(self):
        'Updates indexes after each epoch'
        self.indexes = np.arange(len(self.list_IDs))
        if self.shuffle == True:
            np.random.shuffle(self.indexes)

    def __data_generation(self, list_IDs_temp):
        'Generates data containing batch_size samples' # X : (n_samples, *dim, n_channels)
        # Initialization
        X = np.empty((self.batch_size, *self.dim, self.n_channels))
        y = np.empty((self.batch_size), dtype=float)

        # Generate data
        for i, ID in enumerate(list_IDs_temp):
            # Store sample
            image = np.empty((512, 512, 2))
            sample = self.data_set[ID]
            # read images
            image[:,:,0] = cv2.imread(sample[0], 0)
            image[:,:,1] = cv2.imread(sample[1], 0)
            # save images to array
            X[i,] = image
            # save target values to array
            y[i] = sample[2]

        X = X.astype('float32')
        X/=255
        y = y.astype('float32')
        return X, y
```

Listing A.2: Generator module to load training and validation samples from lookup table

List of Figures

1.1. Superimposition of an aorta model (red) within a fluoroscopic frame. The model is static and acquired preoperatively, while fluoroscopic frames are acquired in real-time during the intervention. Through the superimposition, structures like the aorta are clearly visible and this augmented view can be used to guide a catheter to the correct location for replacing the aortic valve since the location within the vessel is known. The balloon-expandable prosthetic valve is highlighted in the image. 2

3.1. First frame of fluoroscopy run (left) and 30th frame (right). Displacement, as measured by the edge of the diaphragm, is highlighted and is 58 pixels in this example. The rectangular box in the left frame highlight the selected ROI 9

3.2. Functionality of template matching. The template image t is centered to all positions in the image f. A scalar value indicating similarity is calculated based on the template image and the image under the region of the template. This scalar value is written to the corresponding position in an output image. 11

3.3. Heatmap of correlation coefficients with a maximum match at x = 509 and y = 374. 12

3.4. First frame of fluoroscopy run and selected ROI (red frame). The edge of the diaphragm is usually visible in the bottom right of a frame. 13

3.5. Respiratory motion in fluoroscopy run. In this graph, a negative value corresponds to expiration. Since the displacement merely becomes positive, the reference frame is recorded while the patient is in almost full inspiration. 14

3.6. Histogram of samples per displacement of the dataset. Small displacements compose the gross of the dataset, while samples with a displacement over absolute 60 pixels occur rarely. The histogram is symmetric across the y-axis. 17

4.1. The dot product of the filter with every subimage is calculated and creates an output image, also called feature map. Images are taken from [46]. . 20

4.2. Example of max-pooling. The input is divided into four regions sized 2x2 pixels. Passing only the highest value of each region discards 75% of the input. Figure is taken from [45]. 21

4.3. Two-channel-network (left) and siamese network (right). Color code used: cyan = convolution, purple = max-pooling, yellow = fully connected layer. Figure is taken from [48]. 23

4.4. breath_detection repository project structure. 25

4.5. Unoptimized convolutional neural network. It takes a two-channel-image as input and processes it in several convolutional- and pooling layers before a prediction is given by the fully connected layer. 28

4.6. Training- and validation error of the unoptimized network with respect to the number of training epochs. The standard deviation for the last validation run is 12.91 px, which indicates a high spread of data points around the mean. 29

4.7. Visualization of the network's performance on two fluoroscopy runs (a, b) of the test set after 10 epochs of training. The blue curve corresponds to the prediction, the orange curve originates from tracking the diaphragm by using cross-correlation. 29

4.8. Dummy problem to evaluate the network's capabilities. The network is trained to detect the pixel displacement of the white square with respect to a reference frame. 30

4.9. Displacement in y-direction of the white square within the training data. A random shift in x-direction is added in order to make the dummy problem more realistic. 31

4.10. Visualization of the network's performance on the validation dataset after being trained for one (blue) and 10 (orange) epochs. The actual curve of the validation data is additionally drawn green. Evaluation was done for a network with dropout applied (a) and without a dropout layer (b). 31

4.11. Siamese network. Both input images are processed in separate but identical streams. Their results are merged after the last convolutional-/pooling layer. 35

4.12. Mean absolute error of the final network for training set (blue) and validation set (orange) for 10 epochs of training. 36

5.1. Mean absolute error of the final network for training set (blue) and validation set (orange) for 10 epochs of training. The standard deviation for the last validation run is 8.95 px, which indicates high spread of data points around the mean. 38

5.2. Visualization of the network's performance on two randomly picked fluoroscopy runs (a, b) of the test set after 10 epochs of training. The blue curve corresponds to the prediction, the orange curve corresponds to tracking the diaphragm by using the cross-correlation. 38
5.3. Based on the sample (a), a heatmap overlay is created and visualized (b). According to the heatmap, the network pays only minor attention to the diaphragm (bottom right of frames). The focus of the network is mainly on the heart contour. 40
5.4. Comparison of the network's prediction (blue) with the result from cross-correlation (orange). Calculation was done on the fluoroscopy run of the sample in Figure 5.3. 40
5.5. Based on fluoroscopic frame pairs (a), a heatmap overlay is created and visualized (b). The heatmap shows the strong focus the network puts on the diaphragm. 41
5.6. Comparison of the network's prediction (blue) with the result from cross-correlation (orange). The calculation is based on the fluoroscopy run of the same sample shown in Figure 5.5. 42
5.7. Motion prediction of fluoroscopy runs without the diaphragm. The waveform of respiratory motion can be obtained by fitting a polynomial function (blue) that suppresses the cardiac components of higher frequency. . . 43
5.8. Motion prediction of fluoroscopy runs. The diaphragm is not visible at a displacement of below -30 pixels. 43
5.9. Cross-correlation results can vary greatly depending on the ROI. The two selected ROIs in (a) result in curves that have a mean absolute deviation of 5.6 pixels and a maximum absolute deviation of 12.8 pixels with respect to each other. 44
5.10. Cross-correlation results can vary particularly strong if the diaphragm is present on both sides of the image. The two selected ROIs in (a) result in curves that have a mean absolute deviation of 8.1 pixels and a maximum absolute deviation of 22.5 pixels. 45

List of Tables

3.1. Composition of data_set. Most of the available fluoroscopy runs are used for training, while roughly 15% are in the validation or test set. 16

4.1. Mean error of the network with respect to the dummy validation dataset. 32
4.2. Results of a network trained with data that consists of fluoroscopy runs from a different number of patients. The column "Data Increase" lists how much the dataset was increased compared to the dataset in Test 1 and the column "Improvement" lists the percentage error reduction compared with Test 1. 34
4.3. Results of a network trained with data capped at a different number of samples per displacement. The "Improvement" column refers to the validation error reduction compared with Test 1. 34

5.1. Results of analyzing the frame pair with maximum displacement in each validation fluoroscopy run. The column "Absolute Deviation [px]" lists the deviation when measuring the displacement of the diaphragm on its left and right side and calculating the absolute difference of both. The column "Relative Deviation" shows the percentage this deviation forms with respect to the maximum displacement. There is no obvious correlation between deviation and mean validation error. Note that the average mean validation error is smaller than in Figure 5.1 since all runs are weighted equally, regardless of the duration of them. 47

Bibliography

[1] M. Abadi, A. Agarwal, P. Barham, E. Brevdo, Z. Chen, C. Citro, G. S. Corrado, A. Davis, J. Dean, M. Devin, S. Ghemawat, I. Goodfellow, A. Harp, G. Irving, M. Isard, Y. Jia, R. Jozefowicz, L. Kaiser, M. Kudlur, J. Levenberg, D. Mané, R. Monga, S. Moore, D. Murray, C. Olah, M. Schuster, J. Shlens, B. Steiner, I. Sutskever, K. Talwar, P. Tucker, V. Vanhoucke, V. Vasudevan, F. Viégas, O. Vinyals, P. Warden, M. Wattenberg, M. Wicke, Y. Yu, and X. Zheng, "TensorFlow: Large-scale machine learning on heterogeneous systems," 2015, software available from tensorflow.org. [Online]. Available: https://www.tensorflow.org/

[2] P. Ambrosini, D. Ruijters, W. J. Niessen, A. Moelker, and T. van Walsum, *Fully Automatic and Real-Time Catheter Segmentation in X-Ray Fluoroscopy*, 2017, pp. 577–585, exported from https://app.dimensions.ai on 2018/10/25. [Online]. Available: https://app.dimensions.ai/details/publication/pub.1091429767andhttp://arxiv.org/pdf/1707.05137

[3] A. Amidi, S. Amidi, D. Vlachakis, V. Megalooikonomou, N. Paragios, and E. Zacharaki, "Enzynet: enzyme classification using 3d convolutional neural networks on spatial representation," *ArXiv e-prints*, 2017.

[4] S. Amidi. (2017) A detailed example of how to use data generators with keras. [Online]. Available: https://stanford.edu/~shervine/blog/keras-how-to-generate-data-on-the-fly

[5] H. R. Andersen, "Transluminal catheter implanted prosthetic heart valves," *International Journal of Angiology*, vol. 7, no. 2, pp. 102–106, Mar 1998. [Online]. Available: https://doi.org/10.1007/BF01618379

[6] A. V. Arujuna, R. J. Housden, Y. Ma, R. Rajani, G. Gao, N. Nijhof, P. Cathier, R. Bullens, G. Gijsbers, V. Parish, S. Kapetanakis, J. Hancock, C. A. Rinaldi, M. Cooklin, J. Gill, M. Thomas, M. D. O'neill, R. Razavi, and K. S. Rhode, "Novel system for real-time integration of 3-d echocardiography and fluoroscopy for image-guided cardiac interventions: Preclinical validation and clinical feasibility evaluation," *IEEE journal of translational engineering in health and medicine*, vol. 2, pp. 1 900 110; 1 900 110–1 900 110, 01 2014. [Online]. Available: https://www.ncbi.nlm.nih.gov/pubmed/27170872

[7] J. Bergstra, D. Yamins, and D. D. Cox, “Making a science of model search,” *CoRR*, vol. abs/1209.5111, 2012. [Online]. Available: http://arxiv.org/abs/1209.5111

[8] P. Carità, G. Coppola, G. Novo, G. Caccamo, M. Guglielmo, F. Balasus, S. Novo, S. Castrovinci, M. Moscarelli, and E. Corrado, *Aortic stenosis: insights on pathogenesis and clinical implications*, 2016.

[9] P. Y. Carry, P. Baconnier, A. Eberhard, P. Cotte, and G. Benchetrit, “Evaluation of respiratory inductive plethysmography: accuracy for analysis of respiratory waveforms.” *Chest*, vol. 111 4, pp. 910–5, 1997.

[10] F. Chollet *et al.*, “Keras,” https://keras.io, 2015.

[11] ——. (2017) mnist_cnn.py. [Online]. Available: https://github.com/keras-team/keras/blob/master/examples/mnist_cnn.py

[12] A. G. Cribier, “The odyssey of tavr from concept to clinical reality,” *Texas Heart Institute Journal*, vol. 41, no. 2, pp. 125–130, 2014, pMID: 24808769. [Online]. Available: https://doi.org/10.14503/THIJ-14-4137

[13] A. Dosovitskiy, P. Fischer, E. Ilg, P. Häusser, C. Hazırbaş, V. Golkov, P. v.d. Smagt, D. Cremers, and T. Brox, “Flownet: Learning optical flow with convolutional networks,” in *IEEE International Conference on Computer Vision (ICCV)*, 2015. [Online]. Available: http://lmb.informatik.uni-freiburg.de/Publications/2015/DFIB15

[14] eclique, “keras-gradcam,” 2017, [Online; accessed 20-November-2018]. [Online]. Available: https://github.com/eclique/keras-gradcam

[15] A. Geiger, P. Lenz, C. Stiller, and R. Urtasun, “Vision meets robotics: The kitti dataset,” *International Journal of Robotics Research (IJRR)*, 2013.

[16] T. Glasmachers, “Limits of end-to-end learning,” *CoRR*, vol. abs/1704.08305, 2017. [Online]. Available: http://arxiv.org/abs/1704.08305

[17] I. Goodfellow, Y. Bengio, and A. Courville, *Deep Learning*. MIT Press, 2016, http://www.deeplearningbook.org.

[18] jacobgil, “Grad-cam implementation in keras,” 2017, [Online; accessed 20-November-2018]. [Online]. Available: https://github.com/jacobgil/keras-grad-cam

[19] Z. K. M. Joo S. Chuah, “Automatic breath phase detection using only tracheal sounds,” 8 2010. [Online]. Available: https://www.researchgate.net/profile/Zahra_Moussavi/publication/228724229_Automated_respiratory_phase_detection_by_acoustical_means/links/55085b5a0cf27e990e0a83ce/Automated-respiratory-phase-detection-by-acoustical-means.pdf

[20] KDE, *version 4.1.7*. Boston, Massachusetts: Krita Foundation, 2018.

[21] K. Konno and J. Mead, "Measurement of the separate volume changes of rib cage and abdomen during breathing," *Journal of applied physiology*, vol. 22, no. 3, p. 407—422, March 1967. [Online]. Available: https://doi.org/10.1152/jappl.1967.22.3.407

[22] R. Kotikalapudi and contributors, "keras-vis," https://github.com/raghakot/keras-vis, 2017.

[23] A. Krizhevsky, I. Sutskever, and G. E. Hinton, "Imagenet classification with deep convolutional neural networks," in *Advances in Neural Information Processing Systems 25*, F. Pereira, C. J. C. Burges, L. Bottou, and K. Q. Weinberger, Eds. Curran Associates, Inc., 2012, pp. 1097–1105. [Online]. Available: http://papers.nips.cc/paper/4824-imagenet-classification-with-deep-convolutional-neural-networks.pdf

[24] J.-C. Laborde, S. J. Brecker, D. Roy, and M. Jahangiri, "Complications at the time of transcatheter aortic valve implantation," *Methodist DeBakey Cardiovascular Journal*, vol. 8, no. 2, pp. 38–41, Apr-Jun 2012. [Online]. Available: http://www.ncbi.nlm.nih.gov/pmc/articles/PMC3405794/

[25] L. J. Latecki, "Template matching," https://www.inf.u-szeged.hu/~pbalazs/teaching/TemplateMatching.pdf, 1992, accessed: 2018-10-05.

[26] Y. LeCun, B. Boser, J. S. Denker, D. Henderson, R. E. Howard, W. Hubbard, and L. D. Jackel, "Backpropagation applied to handwritten zip code recognition," *Neural Computation*, vol. 1, no. 4, pp. 541–551, Dec 1989.

[27] Y. Lecun, L. Bottou, Y. Bengio, and P. Haffner, "Gradient-based learning applied to document recognition," vol. 86, pp. 2278 – 2324, 12 1998.

[28] J. P. Lewis, "Fast normalized cross-correlation," 1995.

[29] B. D. Lucas and T. Kanade, "An iterative image registration technique with an application to stereo vision," in *Proceedings of the 7th International Joint Conference on Artificial Intelligence - Volume 2*, ser. IJCAI'81. San Francisco, CA, USA: Morgan Kaufmann Publishers Inc., 1981, pp. 674–679. [Online]. Available: http://dl.acm.org/citation.cfm?id=1623264.1623280

[30] MATLAB, *version 9.10.0 (R2017b)*. Natick, Massachusetts: The MathWorks Inc., 2017.

[31] M. A. Mazurowski, M. Buda, A. Saha, and M. R. Bashir, “Deep learning in radiology: an overview of the concepts and a survey of the state of the art,” *CoRR*, vol. abs/1802.08717, 2018. [Online]. Available: http://arxiv.org/abs/1802.08717

[32] T. H. Project, *version 3.3.5*. Boston, Massachusetts: The Horos Project, 2018.

[33] F. G. N. R. Palaniappan, K. Sundaraj, “An overview of breath phase detection – techniques and applications,” *Journal of Telecommunication, Electronic and Computer Engineering*, vol. 10, pp. 33 – 36, 12 2017.

[34] Raghavendra Kotikalapudi , “Visualizing attention on self driving car,” 2017, [Online; accessed 20-November-2018]. [Online]. Available: https://github.com/raghakot/keras-vis/blob/master/applications/self_driving/visualize_attention.ipynb

[35] P. Rajpurkar, J. Irvin, K. Zhu, B. Yang, H. Mehta, T. Duan, D. Ding, A. Bagul, C. Langlotz, K. Shpanskaya, M. P. Lungren, and A. Y. Ng, “Chexnet: Radiologist-level pneumonia detection on chest x-rays with deep learning,” *CoRR*, vol. abs/1711.05225, 2017. [Online]. Available: http://arxiv.org/abs/1711.05225

[36] A. Ratnovsky, D. Elad, and P. Halpern, “Mechanics of respiratory muscles,” *Respiratory Physiology & Neurobiology*, vol. 163, no. 1, pp. 82 – 89, 2008, respiratory Biomechanics. [Online]. Available: http://www.sciencedirect.com/science/article/pii/S1569904808001134

[37] C. J. Schulz, M. Schmitt, D. Böckler, and P. Geisbüsch, “Feasibility and accuracy of fusion imaging during thoracic endovascular aortic repair,” *Journal of Vascular Surgery*, vol. 63, no. 2, pp. 314–322, 2018/10/25 2016. [Online]. Available: https://doi.org/10.1016/j.jvs.2015.08.089

[38] R. R. Selvaraju, A. Das, R. Vedantam, M. Cogswell, D. Parikh, and D. Batra, “Grad-cam: Why did you say that? visual explanations from deep networks via gradient-based localization,” *CoRR*, vol. abs/1610.02391, 2016. [Online]. Available: http://arxiv.org/abs/1610.02391

[39] N. Srivastava, G. Hinton, A. Krizhevsky, I. Sutskever, and R. Salakhutdinov, “Dropout: A simple way to prevent neural networks from overfitting,” *Journal of Machine Learning Research*, vol. 15, pp. 1929–1958, 2014. [Online]. Available: http://jmlr.org/papers/v15/srivastava14a.html

[40] D. Teney and M. Hebert, “Learning to extract motion from videos in convolutional neural networks,” *CoRR*, vol. abs/1601.07532, 2016. [Online]. Available: http://arxiv.org/abs/1601.07532

[41] T. Terunuma, A. Tokui, and T. Sakae, “Novel real-time tumor-contouring method using deep learning to prevent mistracking in x-ray fluoroscopy,” *Radiological physics and technology*, vol. 11, no. 1, pp. 43–53, 2018. [Online]. Available: https://www.ncbi.nlm.nih.gov/pubmed/29285686

[42] M. Unberath, J.-N. Zaech, S. C. Lee, B. Bier, J. Fotouhi, M. Armand, and N. Navab, “DeepDRR–A Catalyst for Machine Learning in Fluoroscopy-guided Procedures,” in *Proc. Medical Image Computing and Computer Assisted Intervention (MICCAI)*. Springer, 2018.

[43] I. Vernikouskaya, W. Rottbauer, B. Gonska, C. Rodewald, J. Seeger, V. Rasche, and J. Wöhrle, “Image-guidance for transcatheter aortic valve implantation (tavi) and cerebral embolic protection,” *International Journal of Cardiology*, vol. 249, 09 2017.

[44] I. Vernikouskaya, W. Rottbauer, J. Seeger, B. Gonska, V. Rasche, and J. Wöhrle, “Patient-specific registration of 3d ct angiography (cta) with x-ray fluoroscopy for image fusion during transcatheter aortic valve implantation (tavi) increases performance of the procedure,” *Clinical Research in Cardiology*, vol. 107, 02 2018.

[45] Wikipedia contributors, “Convolutional neural network — Wikipedia, the free encyclopedia,” 2018, [Online; accessed 21-November-2018]. [Online]. Available: https://en.wikipedia.org/w/index.php?title=Convolutional_neural_network&oldid=869304720

[46] ——, “Sobel operator — Wikipedia, the free encyclopedia,” 2018, [Online; accessed 20-November-2018]. [Online]. Available: https://en.wikipedia.org/w/index.php?title=Sobel_operator&oldid=867844483

[47] W. L. Yaqi Zhang, Billy Wan. (2017) Vehicle motion detection using cnn. [Online]. Available: http://cs231n.stanford.edu/reports/2017/pdfs/625.pdf

[48] S. Zagoruyko and N. Komodakis, “Learning to compare image patches via convolutional neural networks,” *CoRR*, vol. abs/1504.03641, 2015. [Online]. Available: http://arxiv.org/abs/1504.03641

[49] M. Zakkar, A. J. Bryan, and G. D. Angelini, “Aortic stenosis: diagnosis and management,” *BMJ*, vol. 355, 2016. [Online]. Available: https://www.bmj.com/content/355/bmj.i5425

Declaration

I hereby declare that this thesis titled:

Convolutional Neural Networks (CNN)
Applied to Respiratory Motion
Detection in Fluoroscopic Frames

is the product of my own independent work and that I have used no sources or materials other than those specified. The passages taken from other works, either verbatim or paraphrased in the spirit of the original quote, are identified in each individual case by indicating the source. I further declare that all my academic work was written in line with the principles of proper academic research according to the official “Satzung der Universität Ulm zur Sicherung guter wissenschaftlicher Praxis” (University Statute for the Safeguarding of Proper Academic Practice).

Ulm, ...

Christoph Baldauf