MRI cardiac images segmentation robust to motion artefacts

Nathan Galmiche, Nora Vogt, Julien Oster
Laboratoire IADI

October 8, 2022

Summary

1 Introduction

Heart MRI scans are commonly used in the medical field to diagnose different diseases and conditions such as heart failures, damages or defects, valve defects or pericarditis for example. Previous challenges have proven that machine learning can be a great tool to analyse those images. For instance, the ACDC challenge (Automated Cardiac Diagnosis Challenge) had a segmentation task which showed very promising results (0.9 to 0.97 Dice score). However it is often hard to use these tools in a clinical practice as many variations in the imaging settings can happen. Among others, the one that will interest us most is the fact that patients may move during the scan. More specifically, as the heart is in the chest, the simple breathing motion has a big impact on the resulting images. Hence the creation of the CMRx-Motion challenge, where the patients were asked to do different scans where they hold their breath more or less. This resulted in a data set of clean and artefacted images. The challenge proposes two tasks, one classification task where a prediction of the quality of the image is asked and a segmentation task of the left ventricle (LV), right ventricle (RV) and myocardium (MYO). In this paper, we will talk about this second task.

2 Methods

2.1 U-Net

U-net is a convolutional neural network developed in the university of Freiburg in Germany. Its purpose was to be most effective for biomedical image segmentation which made it very interesting for us. Moreover, it has proven its effectiveness during different medical images segmentation challenges.

One of the main advantages this model has is that it stays effective even with little data, one of the major problems of medical imaging. The first part of the model resembles a classic convolutional network with a series of convolutions followed by max poolings. The second part however follows the same scheme but replacing pooling operators with upsampling operators that allow to increase the resolution. As a result, we can compare the architecture of U-net to the shape of a U with a branch that contracts the data and one that expands it. The localization of the expanded data is made possible by using a convolution on the concatenation of the upsampled data with the

features from the corresponding contracted data.

This work uses the basic architecture of U-Net and it was not modified. Of course there are other models deriving from this one, we will mention Nn-UNet further in this report.

2.2 Advchain

Even though U-net is efficient with few images, data augmentation is very important. It is pretty common to use a set of different augmentations and apply them to the images with different parameters. Instead of doing this randomly, Advchain was proposed. It is a framework that uses a pool of four different augmentations and tunes their parameters in order to make it as hard as possible for the model to segment the augmented image.

The four augmentations are :

- an affine transformation that applies a rotation, a translation, a scaling and a shearing. As everybody's body is unique, this allows to recreate varied shapes but also angles in which the image was taken.

- an intensity transformation that applies bias fields on the image. Not only does the intensity vary a lot from one image to another it is also possible that the intensity will vary from one position to another in the image itself. This is imitated by the fields that create inhomogeneities on the image.

- a diffeomorphic transformation that modifies the shape on the image. Much like affine transformations, this translates to the uniqueness of bodies and it allows to teach the model new and varied shapes.

- a noise addition to lower the quality of the image as it can be impacted by movements, equipment or protocol mistakes.

First, these augmentations are randomly chosen and initialized before they are applied to an image. Then, depending on the parameter 'n_iter' chosen by the user, we optimize n_iter times the parameters of the augmentations by maximising a consistency loss function.

Three different options are implemented in Advchain for this loss :

- Kullback-Leibler divergence which compares the probability distributions of the prediction on the original image and the one on the augmented image. It is also called the information gained by choosing the first distribution over the other.

- Mean squared error is the mean of the squares of each pixels from the original and augmented image predictions differences.

- contour loss which compares the edges of the predictions. It is the euclidean distance between the contours of both predictions obtained with a Sobel filter. This loss adds focus to the edges which can quickly be irregular when the model faces a hard case.

Once the augmentations have been optimized, they are applied to the original image and a consistency loss is calculated again to be added to the supervised loss that will be minimized by optimizing the model.

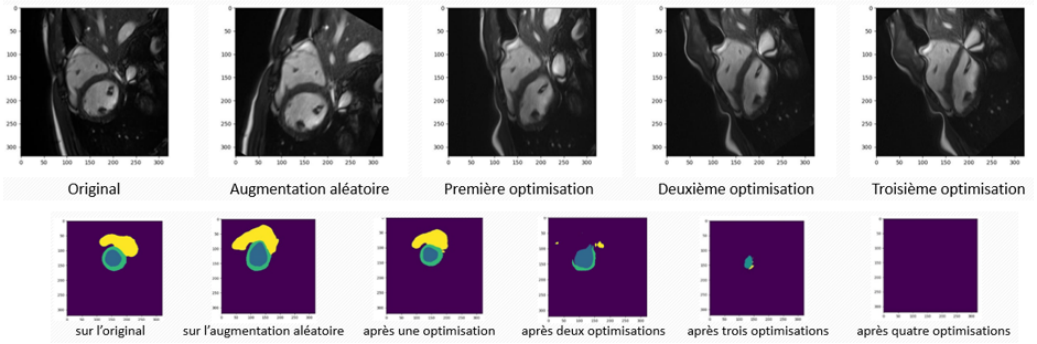

Figure 1: Above : Effect of Advchain on an image from the training set. Under : Effect of Advchain on the model's prediction

2.3 Motion augmentation

The particularity of the challenge and therefore the dataset is that many images were taken under no breath restriction or heavy breathing. this means that many images are made harder for the model due to artefacts that come from the respiratory motions. It was therefore very interesting to augment the images with such artefacts. Moreover, it was interesting to supplement Advchain with an augmentation adapted to the dataset.

To imitate respiratory motion artefacts we adapted a rigid motion simulation from Ziad Al-Haj Hemidi from Lubeck. His idea was to initialize a number of movements depending on parameters and data size. This number dictates how many affine transformations are initialized with each one being tuned in order for it to be close to the previous one. This allows the series of transformations to resemble a real life gradual movement.

To understand why so many transformations are initialized, we have to understand a small part of how the MR image is obtained. The image comes

from a k-space on which a Fourier transform was applied. This k-space is generated slice per slice, intensity per intensity. This means that in order to imitate a motion, each slice of the k-space comes from a position of the body slightly shifted from the previous slice. This explains why and how masks are initialized after the transformations. It is specified in the parameters in which range of the k-space will the transformations be applied. The very first mask hides this range, leaving the rest. As many other masks are then initialized as there are movements. All of them hide almost everything except a slice that is in the range mentioned earlier. The width is random but all the masks have to fit in the range specified in the parameters.

Obviously, each transformation will have a corresponding mask. First the affine transformation is applied to the original image from which we obtain the k-space by applying a reverse Fourier transform. The mask is applied on the k-space and it is added to the final k-space which will group every masked transformed k-space. Once it is complete we apply a Fourier transform on it to obtain an augmented image.

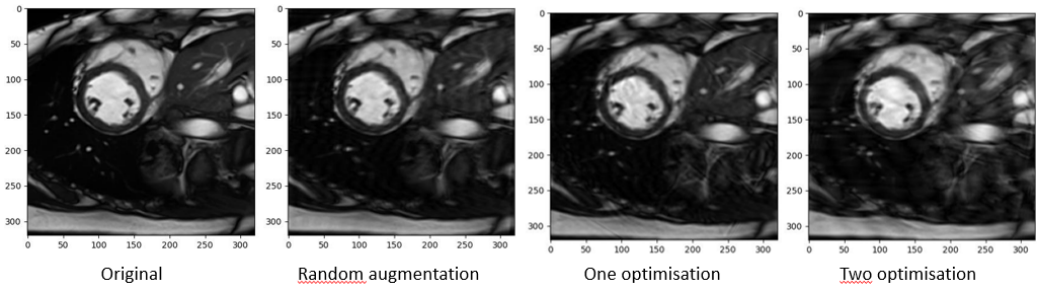

Figure 2: Images from the training set augmenated with a rigid motion simulation.

A respiratory motion is pretty similar to this rigid motion with the difference being that the movement is more periodic. Oksuz' approach uses simple translations that follow a sine wave. A breathing cycle follows one sine period sampled at 256 points to obtain as many translations. Four breathing cycles are applied in the same way as explained before but on the whole image this time.

In practice and for it to work in the Advchain framework, we had to mix both of these ideas. We kept the number of movements and the range as a parameter and changed the affine transformation to only translations in one direction. Their parameters are decided by a sine wave with an amplitude

that is a chosable parameter. The width of the mask is not random anymore so that every slice in the range is the same size. The images produced are slightly different from the rigid motion simulation as we will see in the results section.

3 Experiments

3.1 Dataset

The dataset is split into a training set, a validation set and a testing set. The training set contains images from 20 patients who did each four scans, one where they hold their breath, one where they hold their breath half of the time, one where they breath normally and one where they breath intensely. For each scan both the End Diastolic (ED) and End Systolic (ES) phases are captured. In the end, we have 20*4*2 = 160 images in this set. The other sets were made in the same configuration but with only five patients for the validation set and 20 for the training set. It is important to note that these images are 3D images with sizes ranging from 400*400*9 to 512*512*13.

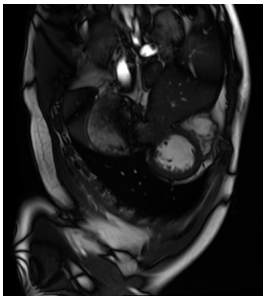

Figure 3: Middle layer of an image from the training dataset

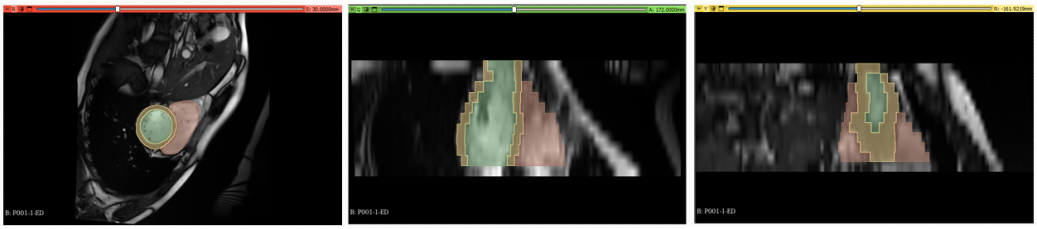

Figure 4: Example of a ground-truth segmentation from different angles

3.2 Metrics

The challenge uses two different metrics to rank participants, the Dice score and the 95% Hausdorff distance. They are both calculated for each class (LV,RV,MYO).

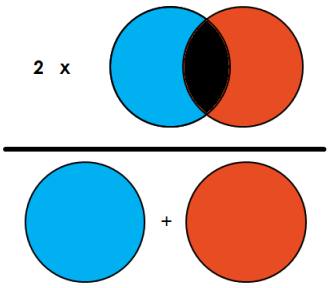

Figure 5: Dice score illustration

The Dice score requires the number of pixel which we can find in both the predicted segmentation and the ground-truth. The score is the double of this number divided by the sum of the number of pixels in the prediction and the ground-truth. This is a proportion which means the best score is one (prediction and ground-truth are equal) and the worst is zero (the prediction has no pixel in common with the ground-truth).

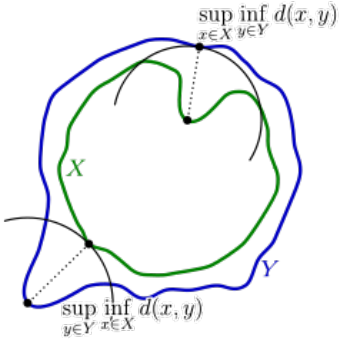

Figure 6: Hausdorff distance illustration

With X the prediction for a class and Y its ground-truth, the Hausdorff distance can be described by :

$$\max(\sup_{y \in Y}(\inf_{x \in X}(d(x,y)), \sup_{x \in X}(\inf_{y \in Y}(d(x,y)))))$$

Using words, we can say that we need to find for both segmentation the point that is farthest in terms of distance to the other segmentation. The score is the maximum of these two distances. This means that the best score is 0 and the worst score depends on the image size but since two pixels on an image cannot be farther than the two opposite corners, we can suppose the maximum Hausdorff distance would be around seven hundred. The challenge uses the 95% Hausdorff distance which simply means that the 5% biggest distances are ignored.

3.3 Network architecture

The part of the network that was the less modified during our experiments is U-Net. We chose to set the number of initial features to 16. We needed a pretty big number so that our model still passed the context information to

the higher resolution layers efficiently. However the idea being to experiment with different tools and set-up, it was interesting to limit the training time so we did not increase this parameter.

As our data are MRI images, the input is characterized by only one channel (gray scale). However we are trying to segment three classes so the output is characterized by four channels.

After testing a few possibilities, we chose to adopt the cross-entropy loss as our supervised loss.

To compare the predictions on the original image and the augmented image in Advchain, we chose to use the mean squared error as well as the contour loss. We simply set the weights to one for the first and 0.5 for the second. This allows to add emphasis on the edges of our predictions as it is often where differences between prediction and ground-truth are. Contour loss is a great way to make predictions more precise but the general shape needs to be set before it is refined. That's why its weight is lighter than the mean square error's.

It is also possible to set a "lazy-load" parameter which, if set to 'True', allows to check if the parameters have already been set. If so, the parameters are not reinitialised. We decided to keep it to False in our experiments to make sure the augmentations are optimised from random augmentations and not previous optimised ones.

Advchain also offers to choose "step_sizes". This is the factor in front of the gradient when optimising the augmentations. It allows to control how much each iteration changes the augmentations. As it is made to be used in many experiments, we want the augmentations to be quickly efficient so we set this parameter to one so that the result is still powerful while staying realistic. This parameter can be adapted depending on the augmentation but we kept it equal for every one of them.

These parameters were adopted for the experiments that we will present in the following parts. Of course some of them could be modified and some choices are very debatable. We will mention some of them in the conclusion as we will talk about unexplored possibilities.

We tested our settings by training using five folds cross-validation on the training set. Also, to obtain predictions on the validation set, we used ensembling of the best models from each of the five folds of the training using the majority voting strategy. It means that for each image, we took the prediction from each folds and kept for each pixel the class that was most often predicted.

4 Results

4.1 Advchain

One very interesting parameter Advchain allows to change is "n_iter". It is the number of times augmentations are optimised before adding the consistency loss to the supervised loss and optimising the model. As we can see in the example in the methods section, maximising the consistency too many times has a drastic impact on the image. We can easily say images get unrealistic with few iterations.

Therefore, we decided to compare different parameters, basing the comparison on a training done without any augmentation. First we tried setting the number of iteration to zero, which means the augmentations are random. Then we tried to use one iteration which is the parameter they use in the original Advchain paper for their experiments. Finally we tried to use three iterations to see what were the effects of using very strong, sometimes unrealistic, augmentations.

	LV - Dice	MYO - Dice	RV - Dice	LV - HD	MYO - HD	RV - HD
No augmentation	0.845	0.716	0.753	7.66	7.39	13.56
Advchain without optimisation	0.860	0.733	0.786	7.38	6.75	9.67
Advchain with one optimisation	0.854	0.735	0.781	7.64	7.47	10.11
Advchain with three optimisations	0.846	0.761	0.791	8.08	8.02	8.69

Table 1: Scores from our experiments with Advchain. Using the challenge's metrics

The first expectation we find validated is that adding augmentations through Advchain improves the results we had originally obtained. This is true whatever the parameters we use which was expected since we only added a framework, keeping the same base. However, it is harder to compare the different parameters we tried as the scores are pretty close. It seems the scores that are closest are the random augmentations and augmentations

with one iteration of optimisation. Augmenting with three augmentations seems to return more drastic results, some classes are improved while others are decreased. This could be explained by the fact that the augmentations are strong which makes it robust to strong variations but it could be too much, making some classes unrecognizable by the model, hence it does not learn from these images.

Thus we decided to use one iteration of optimisation even though using three could improve some classes, it costed other classes' precision and we found that the results were more often worst than better. Optimizing once was still interesting because other parameters could vary and it is interesting to see the impact it will have on the new augmentation we will add.

4.2 Respiratory motion augmentation

As you can see from the images in the methods section, our first shot with the rigid motion was already pretty promising. We then focused our experiments on the respiratory motion. As we described the new augmentation in the methods section, the two parameters we tuned are the amplitude of the translations and the number of movements. The first one decides how much the sine is multiplied by to obtain the parameters of the translations applied on the image. The second one is the number of slices we take from the k-space in order to modify them.

First we tried to use a small number of movements by setting the parameter to five. We found that the effect was too strong as the ghosts are too few and too visible. As it was not realistic, we tried to use fifteen movements. The results looks more interesting so we tried to push it to seventy. The difference between those last two parameters is not so obvious but seventy movements makes it look more like a blur and fifteen looked a little stronger so we decided to use fifteen movements for the rest of the experiments.

Then we tried different amplitudes, starting with an amplitude of one which, as we can expect, is too low to see a significant change on the image. An amplitude of three starts to show some ghosts but as it is an augmentation to challenge the model, we need to make it pretty strong. Five looked the most realistic while being strong enough as ten put the ghosts too far apart from each other making the augmentation look less realistic. As we settled for the amplitude of five with fifteen movements, we tried to see the effect adding the augmentation had on our scores.

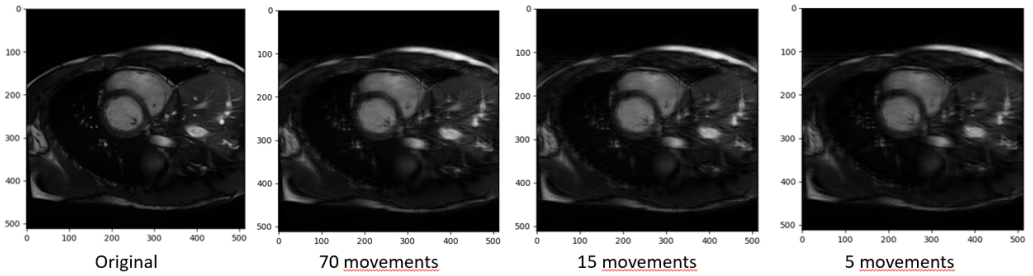

Figure 7: Images from the training set augmented with a respiratory motion simulation with different numbers of movements.

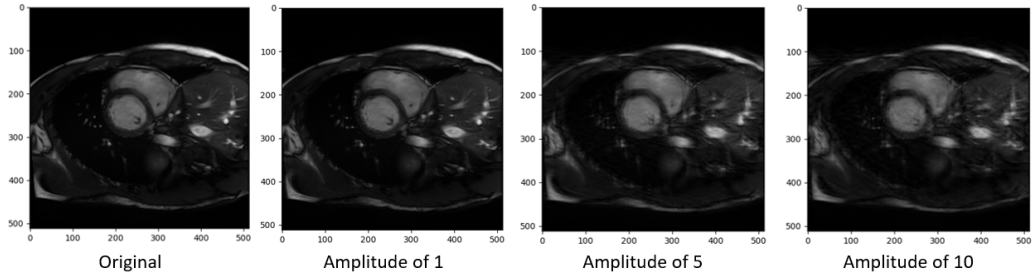

Figure 8: Images from the training set augmented with a respiratory motion simulation with different amplitudes of translations.

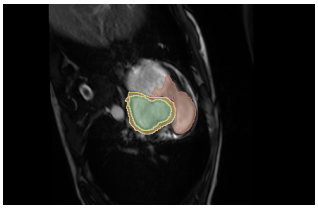

Figure 9: Example of a poorly predicted right ventricle on an image from the validation set.

The first thing we note (Table 2) is that the scores for the left ventricle and its myocardium are very close so the augmentations did not really improve this point. However, there seem to be a promising improvement on the right ventricle, especially with the respiratory motion augmentation as the Hausdorff distance is improved by about two pixels. This is interesting because this class is the hardest one to recognize for the model as it is harder to recognize on the extreme parts of the heart. The bottom and top slices' classes were often poorly predicted by the model, especially the right ventricle's.

	LV - Dice	MYO - Dice	RV - Dice	LV - HD	MYO - HD	RV - HD
Original Advchain	0.860	0.733	0.787	7.38	6.75	9.67
Advchain + rigid motion	0.859	0.731	0.7956	6.89	7.01	9.72
Advchain + respiratory motion	0.854	0.727	0.799	7.54	6.94	7.68

Table 2: Scores from our experiments with Advchain and the motions augmentations added. Using the challenge's metrics

5 Conclusion

Our results seem pretty promising as Advchain improved our results and adding the respiratory motion augmentation seems to have an effect at least on the most difficult class as right ventricle's score improved with our last augmentation. Moreover, many optimisations and improvements can still be made. Firstly on the motion augmentation that we tested with few movements to keep a short training time. It could be interesting to see how changing the number of movements and also where we apply them in the k-space impacts the realism of our augmentation. Also, the paper we used to make our augmentation more like respiratory motion used a repetition of the movements so instead of using one sine period we could use more and modify more parts of the k-space or as many parts but larger. The transformations applied to the k-space are translations so it would make sense to use elastic transformations instead as it might be more realistic in the case of breathing.

Secondly, we only used one augmentation for every image used in the training, we have not mentioned trying to use more because it first had no impact and it is a track we put aside until now. It could be worth trying to investigate why scores did not improve in order to make it effective.

Thirdly, the loss function we used during training was a cross-entropy loss as some initial results indicated we should use. Some later ones showed that other losses would also be interesting to try such as the dice loss. We also have not tried to use the contour loss or other losses that would be worth trying.

Fourthly, the respiratory motion augmentation was incorporated into Ad-

vchain but after modifying it a few times, the impact of the optimisation is almost null. In fact, only the amplitude of the translation are optimised. Other parameters could be optimised such as the number of movements or even what part of the k-space is modified.

Finally, we found that using Nn-Unet was way more effective than U-Net. In order to be able to compete with the top scores, we would have to use Nn-Unet and it would be really interesting to see how our implementation of Advchain with the respiratory motion augmentation reacts with this model.

To conclude we find that U-net is a performing model but Nn-Unet showed its superiority. It would thus be interesting to add Advchain to it with the respiratory motion augmentation and see how it fares compared to the other participants of the challenge. More so since so many aspects of our implementation are improvable, looking at the pretty promising results we already obtained gives hope that we could improve our scores and be among the best.

6 References

Olaf Ronneberger, Philipp Fischer, and Thomas Brox: U-Net: Convolutional Networks for Biomedical Image Segmentation

Chen Chena, Chen Qinb, Cheng Ouyanga, Zeju Lia, Shuo Wangc, Huaqi Qiua, Liang Chena, Giacomo Tarronia, Wenjia Baia, Daniel Rueckert: Enhancing MR Image Segmentation with Realistic Adversarial Data Augmentation

Ilkay Oksuz, Bram Ruijsink, Esther Puyol-Antón, James R. Clough, Gastao Cruz, Aurelien Bustin , Claudia Prieto, Rene Botnar, Daniel Rueckert, Julia A. Schnabel, Andrew P. King: Automatic CNN-based detection of cardiac MR motion artefacts using k-space data augmentation and curriculum learning

Fabian Isensee, Jens Petersen, Andre Klein, David Zimmerer, Paul F. Jaeger, Simon Kohl, Jakob Wasserthal, Gregor Köhler, Tobias Norajitra, Sebastian Wirkert, and Klaus H. Maier-Hein: nnU-Net: Self-adapting Framework for U-Net-Based Medical Image Segmentation