Deep Learning Techniques for N-MRI Image Acquisition and Motion Noise Removal

Kevin Munda

Abstract

Patient motion during the acquisition of magnetic resonance images (MRI) is an important issue for this imaging technique: it could lead to unwanted artefacts in the MR scan that can affect the interpretability of the acquired image, possibly leading a wrong medical analysis. In this work, we devised a method to recover a clear image, starting from a motion-corrupted one, through the use of machine learning algorithms such as convolutional neural networks (CNNs). Motion-corrupted images are obtained starting from a dataset of real and clean scans in which motion is simulated as a sequence of random 3D rigid movements, combined together in the k-space to obtain the final corrupted image. Augmenting the dataset with artefacted data is useful to train a model that can generalise better and be robust even in presence of real artefacts. The implemented neural network is able to correct motion artefacts.

I. Introduction

Magnetic resonance imaging (MRI) is a safe medical technique used to acquire pictures of the anatomical structure and physiological processes of the human body. MRI scanners use strong magnetic fields, measuring the field gradients and signal intensity, to generate images of the organs in the body. MRI is widely used in hospitals and clinics for medical diagnosis, staging and follow-up of diseases (e.g. tumors) without exposing the body to ionizing radiations. Due to the nature of the related applications, it is of crucial importance to obtain very clear images. However the long data acquisition time required by MRI causes many difficulties during its clinical application. Patient movements during the acquisition of MRI can cause unwanted image artefacts, which can be originated from two different type of motion: rigid motion, i.e. movement of some rigid part of the body (e.g. a movement of the head); non-rigid motion, i.e. movement of some non-rigid part of the body (e.g. arterial pulsation or motion due to breathing). These artefacts can affect the interpretability of the acquired MR image, leading to possibly wrong medical analysis by the doctor or to increasing costs, in the case in which the scan must be discarded and the data acquisition repeated. The main goal of this work is to show how, through the use of convolutional neural networks, is possible to recover most of the information from an image with motion artefacts. The paper will follow this structure: section II is about the current state of the art; in section III is explained the motion artefact model used to simulate motion artefacts inside clean scans; in section

IV the dataset used for the experiments is described; in section V is analysed the neural network architecture used to perform motion correction; section VI explains the choices made for the training of the model; section VII shows the most relevant results obtained; section VIII is a brief conclusion about this work with possible future improvements.

II. Related Works

The problem of motion removal in MR images has been recently addressed in a new way with the use of deep learning algorithms. Generally the application of motion correction techniques is performed during or after the acquisition of MR scans. The approaches here cited are all retrospective. Pawar *et al.* [1] used a U-Net like encoder-decoder architecture to perform motion correction on simulated and real artefacted data; in particular the neural network was trained to suppress motion artefacts from individual 2D axial slices of MR scans. Their approach worked well, even if they considered a quite limited range of motion: in fact they simulated 3D rigid body random motion between $\pm 5mm$ of translation and $\pm 5°$of rotation on all three axis. Shaw *et al.* [2] used a convolutional neural network, the HighRes3DNet architecture, to remove artefacts from 3D image volumes, using as input to the network simulated artefacted images; they tested the network also on real-world data, obtaining satisfying results. Duffy *et al.* [3] employed a modified HighRes3DNet architecture and a HighRes3DNetGAN to perform motion correction. The networks were trained using 3D voxels with simulated motion artefacts. From image inspection and analysis of the results, they found that the HR3DNet was able to eliminate a large portion of the ghosting artefact preserving the sharp tissue boundaries, while the HR3DNetGAN obtained sharper tissue contrast, however it tended to over-enhance edges and regions with residual artifacts. Usman *et al.* [4] used a GAN [5] to perform motion correction. In their work they employed standard CG-SENSE to reconstruct corrupted k-space data, in order to obtain motion-corrupted images; then these images are fed to the network for the cleaning of motion artefacts. Their study focuses also on the influence of some parameters in the reconstruction of clean MR scans, in particular they analysed the effect of the amount of motion, the influence of the number of shots and the influence of the encoding trajectory. They showed that their approach is quite robust against these parameters. The work of Ghodrati *et al.* [6] focuses on comparing the performances of two different neural network architectures using four different loss functions: the two networks implemented are a Unet and a residual network (ResNet) while the loss functions are the pixel-wise L1, the pixel-wise L2, the structural dissimilarity (Dssim) and the perceptual loss. The network was trained and validated with two datasets obtained from real patients. The authors concluded that the network trained with the perceptual loss was the one able to obtain the better results compared to the other loss functions, from an image quality point of view. Moreover, the ResNet generates reconstructed images with a similar quality compared to the ones generated by Unet, with the advantage that ResNet requires the 8% of the trainable parameters needed for Unet. In this work, we combine some of the approaches previously described in order to obtain the best model that solves our problem.

III. Motion Artefact Model

The type of motion artefacts depends on the amount and timing of patient movement as revealed in the k-space,that is the space in which MRI imaging data is acquired. K-space data is related to image data by the 2D or 3D Fourier transform. In practice, k-space often refers to the temporary image space (usually a matrix) in which data from digitized MR signals are stored during data acquisition. Movements at the k-space centre correspond to low image frequencies and result in ghosting artefacts, where the image is repeated. These kind of artefacts are caused by quasi-periodic motion e.g. respiration. Motion occurring in the outer regions of the k-space, corresponding to the acquisition of high image frequencies, manifested as ringing artefact in the reconstructed image. The most commonly observed motion artefacts result in minor blurring due to small movements during k-space acquisition. Additionally, the k-space scanning trajectory and whether the acquisition is performed in 2D or 3D influence the motion artefacts inside the scan. In literature there are several ways to simulate motion artefacts to be added to a clean scan. In this work the procedure used to add such an effect to the images is the same used in [2] and depicted in Fig. 1. The procedure consists of five steps, each one analysed in the following. The first step is to generate a random movement model by sampling movements from different probability distribution functions: each movement is modelled by a 3D affine matrix A comprising a rigid 3D rotation and translation in the image domain, where the angles of rotation are randomly sampled between (-10°, 10°) and the translation between (-10mm, 10mm) in all three axis. Poisson distributions are used to sample the magnitude of rotation and translation of each of the N movements, with the assumption that small movements occur more often and large movements less frequently, while a uniform distribution is used to sample the time t in k-space at which each movement occurs. The sequence of movement transforms $\{A\}_{i=0}^{N}$ is combined incrementally in log-Euclidean space [7] using the matrix exponential $exp_M(A) = \sum_{k=0}^{\infty} \frac{A^k}{k!}$ and the corresponding matrix logarithm. By transferring to the log-Euclidean domain is possible to create weighted combinations of transformations and to linearly interpolate between them. The second step is to "demean" the movements: in order to keep in the same position the barycenter of the 3D volume after the application of the augmentation model we have to "demean" each affine transform A_i by pre-multiplying by the inverse of the average transform A_{avg}, computed as the weighted sum of the sequence of N affine transformations in log-Euclidean space:

$$A_{avg} = exp_M(\sum_{i=0}^{N} \hat{w}_i \log_M(A_i)) \tag{1}$$

where $\hat{w}_i$ is a weighting given to the i-th movement. Each A_i is weighted by its signal contribution to the final image. This means that movements at the k-space centre (low frequencies) have a higher weight. Each weight is estimated by masking the 3D k-space of the clean MR volume I_0 with a binary mask M_i corresponding to the k-space elements of the i-th movement, transforming back to the image domain, and summing the resulting voxel intensities, with the weights then normalised to sum to 1. The third step is to apply each "demeaned" affine transform to the original artefact-free

image volume I_0 and resample using b-spline interpolation. After each transformation, the 3D Fourier transform of each resampled image is computed.

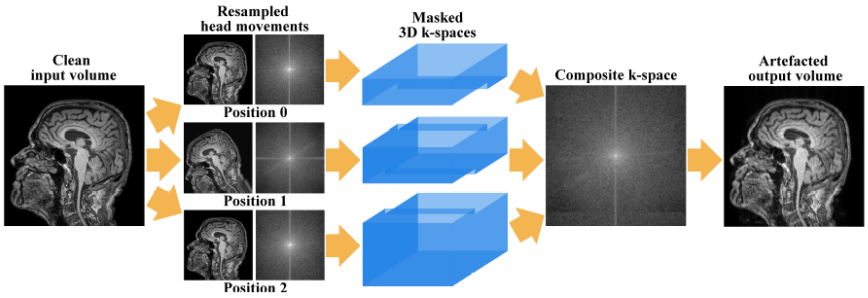

Fig. 1 Motion Artefact Model from [2]

The fourth step is to combine the 3D Fourier transforms in the sequence, joined together at sampled time t, obtaining a complete k-space of the scan with movement. In the fifth and final step the inverse 3D Fourier transform of the composite k-space is derived and the magnitude image provides the final artefacted sample.

IV. Dataset

The dataset used in this work is composed by 576 MRI scans of healthy patients, each one with size 180x240x241. The datasets used for training and testing the deep learning model are composed by 2D images with size 240x176 derived by the original volumes taking for each MRI scan the central slices, which are the ones that show clearly the anatomical parts of the patient, along the three directions x,y and z. An example of a batch of 3 images, one for each direction, is showed in Fig. 2.

(a) **(b)**

Fig. 2 Examples of images inside the sliced datasets. (a) Batch of 3 slices along x,y and z taken from clean scans (b) Batch of 3 slices along x,y and z taken from motion-corrupted scans

Each dataset is composed by two sets of images: the motion-corrupted ones and the original ones. In particular the training set is composed by two sets, each one with 461 batches, for a total of 1383 images, the validation set is composed by two sets, each one with 104 batches, for a total of 312 images, and the test set is composed by two sets, each one with 11 batches, for a total of 33 images.

V. Neural Network Architecture

The single pre-processing applied consists of reshaping images to have all the same shape (240, 176) and normalization of pixel values between 0 and 1. The architecture used in this work is an autoencoder, with a U-Net like structure, quite similar to the one used in [4]. This architecture involves an encoder and a decoder: the encoder learn to compress the relevant information from the corrupted images discarding the corruption such that the decoder is able to recover a clean counterpart, free from motion artefacts. The high level structure of the autoencoder is showed in Fig. 3:

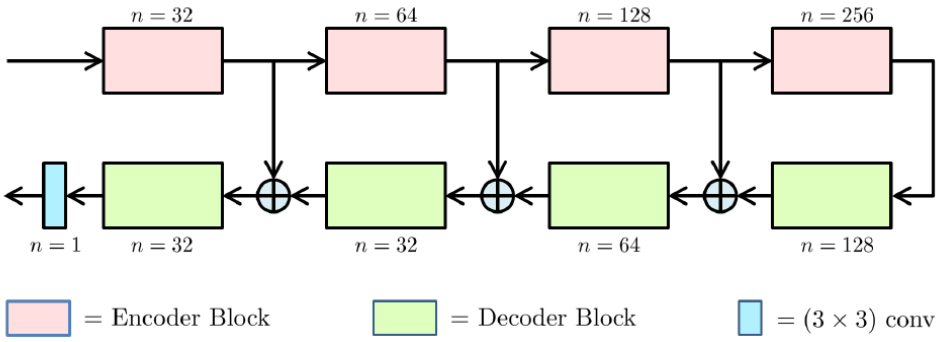

Fig. 3 Autoencoder structure with skip connections taken from [4]

In this structure there are also skip connections from encoder blocks to decoder blocks: these are useful to recover more details for a better image restoration. In fact the encoder learns to compress image into high-level features necessary for image restoration, but it may remove some details along with the corruption, while, with skip connections, low-level features are transfered from the encoding path to the decoding path to recover fine details. Additionally, also residual connections are implemented in this model inside each encoder/decoder block as showed in Fig. 4 and Fig. 5. These techniques allow an efficient gradient back-propagation, avoiding problems such as vanishing gradients and slow convergence [8]. The last layer in the high-level structure is a Convolutional layer with one single feature map, with kernel of dimension (1, 1) and with "sigmoid" as activation function. Each encoder block has the same structure, showed in Fig. 4: it is composed by five Convolutional layers, each one with n feature maps (the number of feature maps for each block is specified in Fig. 3), except for the middle layer that has $\frac{n}{2}$ feature maps. Padding is used to keep the dimension of feature maps same inside each block. Strides is set to 1 for all layers except for the first one, that is set equal to 2, in order to down-sample the dimension of the feature maps. A residual connection is implemented between the output of the first layer and the input of the last layer. Moreover, each Convolutional layer is followed by a Batch Normalization layer and an Activation layer, with "ReLU" as non-linear activation function. The decoder blocks, whose architecture is showed in Fig. 5, have the same structure but with some differences: Convolutional layers are replaced by Transposed Convolutional layers; stride is equal to 2 in the last layer instead of the first one, in order to up-sample the feature maps; after each Activation layer there is a Dropout layer, with dropout equal to 0.2, used to avoid that the

network overfits over the training set and so have also a better performance with unseen images. Additionally, in some experiments, i have tested also dilated convolutions in the encoder/decoder blocks. In particular the sequence of dilated convolutions applied to the encoders is: (1, 1) for the first layer, (2, 2) for the second and third layer, (4, 4) for the fourth and fifth layer. The sequence of dilated convolutions applied to the decoders is the same but in reverse order.

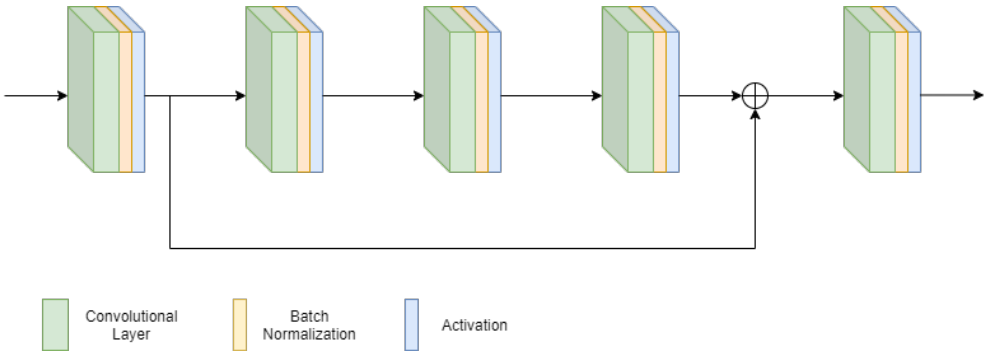

Fig. 4 Encoder structure. Each layer has n feature maps except for the middle layer, that has $\frac{n}{2}$ feature maps. Strides is set equal to 1 in all layers, except for the first one in which is equal to 2. All kernels have dimensions (3, 3).

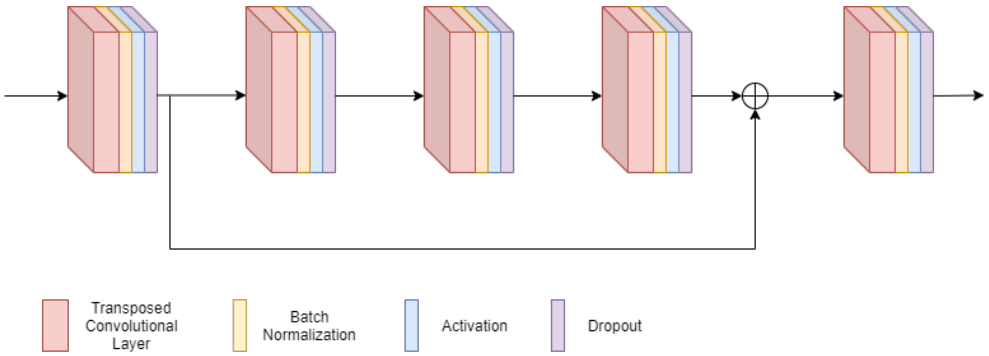

Fig. 5 Decoder structure. Same structure of the encoder, but with Dropout layers after the Activation layer. Strides is set equal to 2 in the last layer.

VI. Model Training

The model was trained for 300 epochs with learning rate equal to $\lambda = 0.0001$ and with batch size equal to 1, with each batch containing 3 images. The optimizer used is the RMSProp, which is the adaptation of the Rprop algorithm for mini-batch learning. Rprop [9] combines the idea of using the sign of the gradient with the idea of adapting the step size individually for each weight. However Rprop doesn't work well when working with mini-batches due to the fact that in this case we divide by different gradient every time. The central idea of RMSProp is to keep a moving average of the squared gradients for each weight, defined in equation (2), and then divide the gradient by the square root of this average when update the weights, as in equation (3).

$$E[g^2]_t = \beta E[g^2]_{t-1} + (1-\beta)g_t^2 \tag{2}$$

$$w_t = w_{t-1} - \frac{\eta}{\sqrt{E[g^2]_t}} g_t \tag{3}$$

where $g = \frac{\delta C}{\delta w}$ is the gradient of the cost function with respect to the weight w. The default value for the parameter β is 0.9. The loss functions used in this work are two: the Mean Squared Error (MSE), used also as a metric to evaluate the performance of the network, and the Perceptual Loss. I decided to test the use of the Perceptual Loss after the results obtained in [6], in which the authors show that the images obtained with this loss have a better quality compared to images reconstructed using other types of loss, even if the values obtained for the metrics used to evaluate quantitatively the correction of motion are lower with respect to the values obtained with the other losses. Mathematically the two losses are defined in the following equations:

$$MSE = \frac{1}{n}\sum_{i=1}^{n}(Y_i - \hat{Y}_i)^2 \tag{4}$$

$$L_{percep} = 0.65L_{percep}(VGG_1) + 0.3L_{percep}(VGG_2) + 0.05L_{percep}(VGG_3) \tag{5}$$

where, in equation (4), Y_i and $\hat{Y}_i$ are respectively the original image and the corrected image, while in equation (5) $L_{percep}(VGG_i)$ represents the Euclidean distance between the produced features of imported Y and $\hat{Y}$ to VGG16 in the first layer of the i^{th} block after activation. The structure of the perceptual network is showed in Fig. 6 taken from [6]:

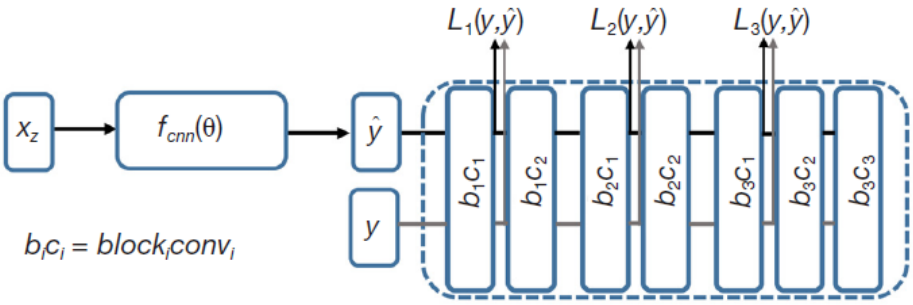

Fig. 6 Perceptual loss network: VGG16 pre-trained network used as a perceptual loss network.

At each epoch, the VGG-16 network receives two independt inputs, the corrected image $\hat{y}$ obtained from the neural network $f_{cnn}(\theta)$ and the original image y, and the optimizer tries to minimize the weighted perceptual loss defined in Eq. (3). A higher weight is assigned to the first layer of the VGG-16 because is assumed that the exctrated features

become gradually more abstract in the deeper layers. The final number of epochs was defined after several experiments during which i observed always the same behaviour of the network: at some point, regardless on which loss function is used, the validation loss continues to oscillate around a certain value, so it stops decreasing, while at the same time, the training loss continues to decrease, obtaining better results for images in the training set but no for images in the validation set. Due to this oscillation of the validation loss, in order to avoid the overfitting of the network, the training was stopped at 300 epochs, when the correction of the motion in the images of the training set was already good, even because much longer trainings (about 900 epochs) were tested, searching for a phase transition in which the validation loss jumps to lower values, but no significant result in that sense was obtained. As we will see in section VII, the fact that, at some point, the validation loss stops decreasing will affect the results of the motion-corrected images. If the training goes beyond the 300 epochs, the quality of the results reached in correcting the images of the training set increases constantly, obtaining as output corrected images with sharp edges and with more details recovered, compared to the results obtained when the input of the network is an image of the validation or test set. This is a useful result when the task is a retrospective correction of motion artefacts in MR scan, in fact in this way is possible to recover a posteriori a very clean image starting from the motion-corrupted one. However one of the goal of this work is to obtain an algorithm as much generalized as possible in order to work also on unseen images, for other possible applications, like for example motion-correction in real time.

VII. Results

In this section are showed the results obtained with the techniques previously described. In particular here are introduced the metrics used to evaluate qualitatively the results. Four are the quantities considered to define how the model performs: the Mean Squared Error (MSE) defined in Eq. (4), the Structural Similarity Index (SSIM) defined in Eq. (6), the Signal to Noise Ratio (SNR) defined in Eq. (7) and the Mean Absolute Percentage Error (MAPE) defined in Eq. (8).

$$SSIM(y, \hat{y}) = \frac{(2\mu_{\hat{y}}\mu_y + C1)(2\sigma_{y\hat{y}} + C2)}{(\mu_{\hat{y}}^2 + \mu_y^2 + C1)(\sigma_y^2 + \sigma_{\hat{y}}^2 + C2)} \tag{6}$$

$$SNR(y, \hat{y}) = 20 \log_{10} \left[\frac{mean(\hat{y}^2)}{mean(y - \hat{y})^2} \right] \tag{7}$$

$$MAPE = \frac{100\%}{n} \sum_{t=1}^{n} \left| \frac{y_t - \hat{y}_t}{y_t} \right| \tag{8}$$

where $\mu_{\hat{y}}$, μ_y, $\sigma_{\hat{y}}$, σ_y and $\sigma_{\hat{y}y}$ are respectively the local means, standard deviations and cross-covariance for corrected and original images. $C1$ and $C2$ are two variables used to stabilize the division with weak denominator. First are presented the results obtained without dilated convolutions. In Fig. 7 and Fig. 8 we can see the outcomes of the motion correction using the two loss functions described in Sec. VI respectively for a sample taken from the train set and a sample taken from the test set. In both cases, the reconstruction performed is quite good, with similar results obtained for both MSE and Perceptual loss, even if the images obtained with the latter show a slightly higher resolution, with sharper edges and more details recovered. As Fig. 8 shows, the network is able to recover information also from unseen images with good results, even if with a lower quality with respect to the reconstructed images of the training set. The metric results obtained in this case are shown in Tab. 1, 2, 3 and 4, concerning the training set, and in Tab. 5, 6, 7 and 8, concerning the test set: the values shown are obtained comparing the different sets of images (i.e the corrupted train/test set, the corrected train/test set using MSE and the corrected train/test set using Perceptual loss) with respect to the train/test set containing the original images. In each table there are the results evaluated along each axis and also over the whole set. Comparing the metrics, is possible to see that the differences between the use of MSE and Perceptual loss are quite small: the use of the Perceptual loss leads to slightly better results in particular over the training set, while MSE loss leads to slightly better results over the test set. However, as said before, even if the metrics values are slower, the network trained with the Perceptual loss achieves images with an higher image quality.

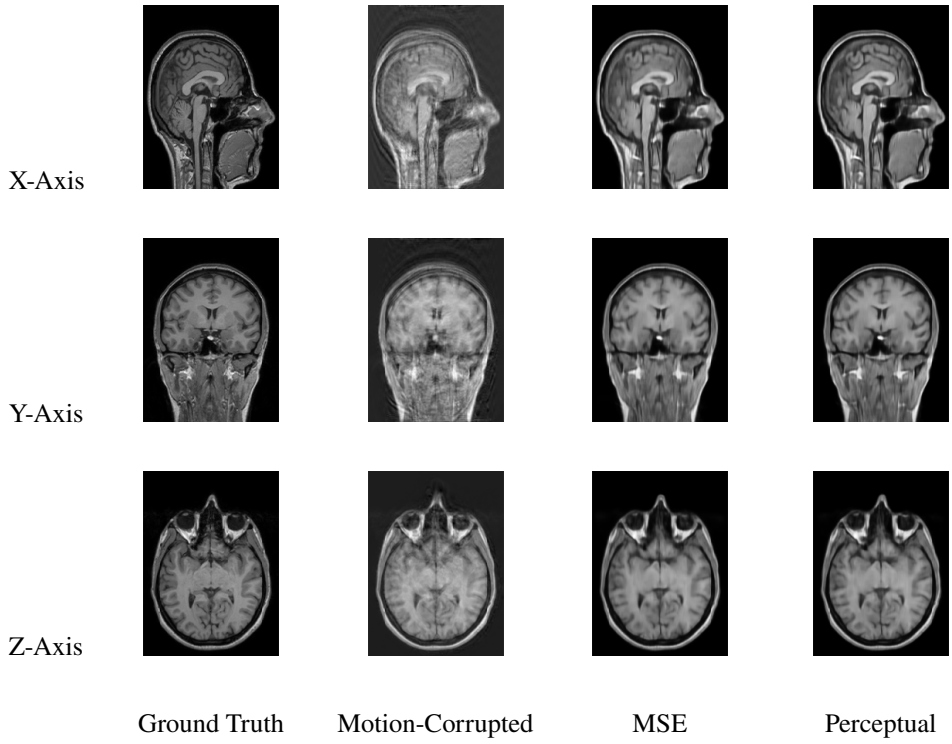

Fig. 7 Motion Correction without dilated convolutions. The sample is taken from the training set.

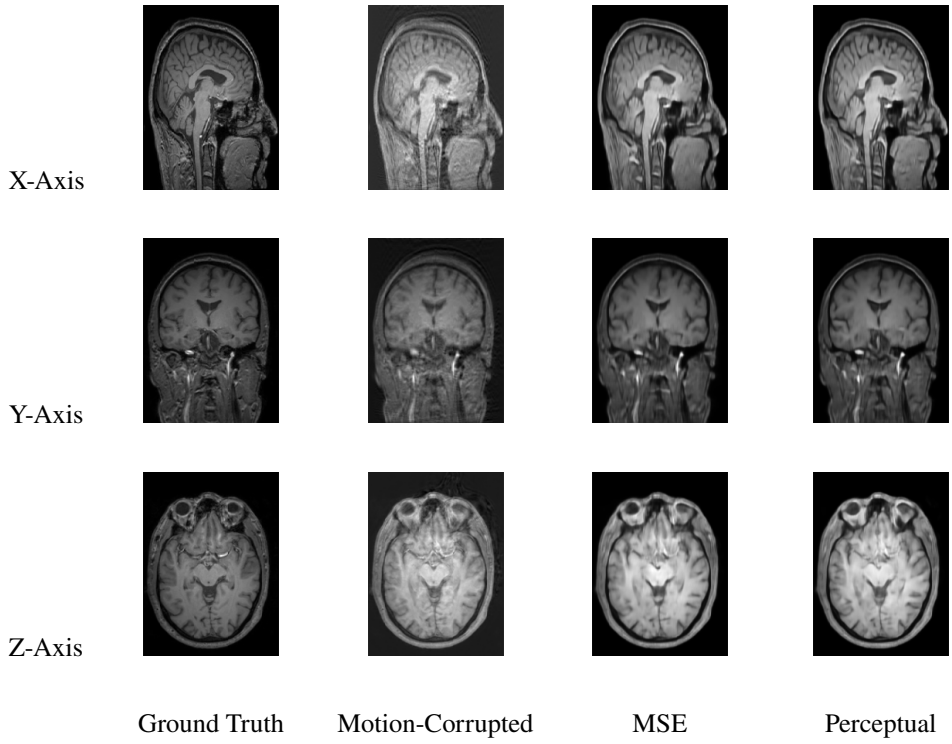

Fig. 8 Motion Correction without dilated convolutions. The sample is taken from the test set.

Mean Squared Error			
Axis	Corrupted	MSE	Perceptual
x	0.0279	0.00172	0.00164
y	0.0256	0.00144	0.00138
z	0.0165	0.00124	0.00124
Tot.	0.0233	0.00147	0.00142

Table 1 MSE results for the network without dilated convolutions applied on the training set

MAPE			
Axis	Corrupted	MSE	Perceptual
x	15.32%	3.15%	3.15%
y	14.45%	3.08%	3.13%
z	11.27%	2.82%	3%
Tot.	13.68%	3.019%	3.1%

Table 2 MAPE results for the network without dilated convolutions applied on the training set

SSIM			
Axis	Corrupted	MSE	Perceptual
x	0.404	0.828	0.837
y	0.435	0.825	0.827
z	0.477	0.854	0.853
Tot.	0.439	0.836	0.839

Table 3 SSIM results for the network without dilated convolutions applied on the training set

SNR			
Axis	Corrupted	MSE	Perceptual
x	15.83	27.95	28.19
y	16.40	28.71	28.96
z	18.73	29.36	29.38
Tot.	16.98	28.68	28.84

Table 4 SNR results for the network without dilated convolutions applied on the training set

Mean Squared Error			
Axis	Corrupted	MSE	Perceptual
x	0.0283	0.00343	0.00355
y	0.0253	0.00328	0.00333
z	0.0231	0.00372	0.00400
Tot.	0.0256	0.00348	0.00363

Table 5 MSE results for the network without dilated convolutions applied on the test set

MAPE			
Axis	Corrupted	MSE	Perceptual
x	15.35%	3.87%	3.9%
y	14.39%	3.58%	3.63%
z	13.76%	4.18%	4.15%
Tot.	14.5%	3.88%	3.89%

Table 6 MAPE results for the network without dilated convolutions applied on the test set

SSIM			
Axis	Corrupted	MSE	Perceptual
x	0.421	0.758	0.756
y	0.445	0.748	0.745
z	0.479	0.788	0.785
Tot.	0.448	0.765	0.762

Table 7 SSIM results for the network without dilated convolutions applied on the test set

SNR			
Axis	Corrupted	MSE	Perceptual
x	15.73	24.73	24.61
y	16.33	25.18	25.17
z	16.82	24.43	24.19
Tot.	16.29	24.78	24.66

Table 8 SNR results for the network without dilated convolutions applied on the test set

Now are shown the results obtained when dilated convolutions are applied. In Fig. 9 and Fig. 10 there is an example of correction using the same samples as before. From the images we can see that the reconstruction happens, however, observing carefully, the images are not clear as before, some details are more blurred with respect to the previous case,

leading to corrected images with a lower quality compared to the ones obtained in Fig. 7 and Fig. 8.

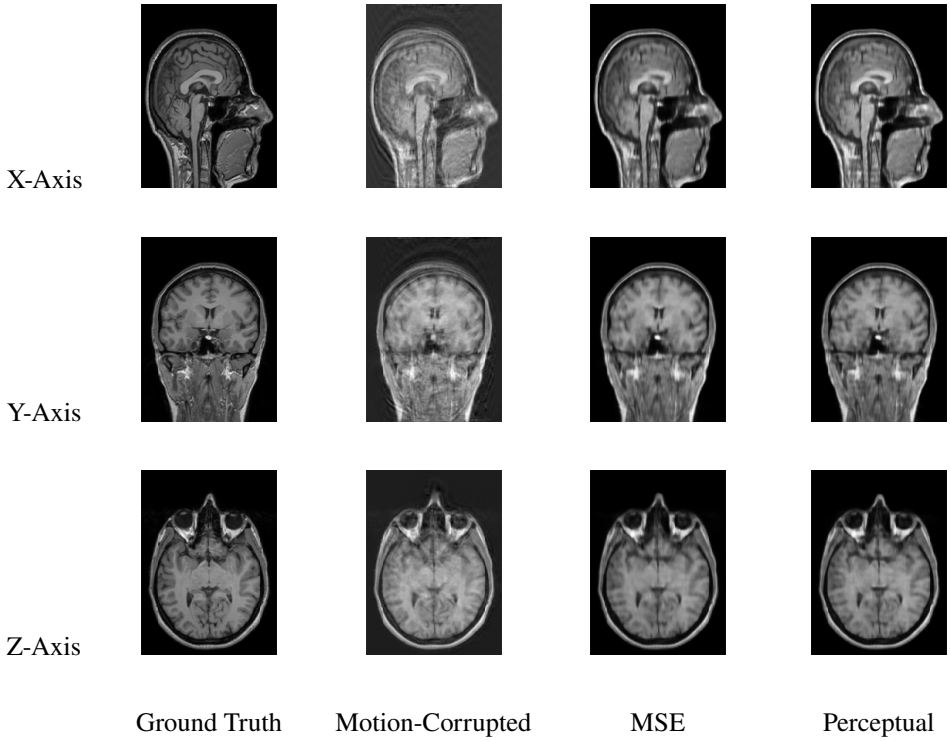

Fig. 9 Motion Correction with dilated convolutions. The sample is taken from the training set.

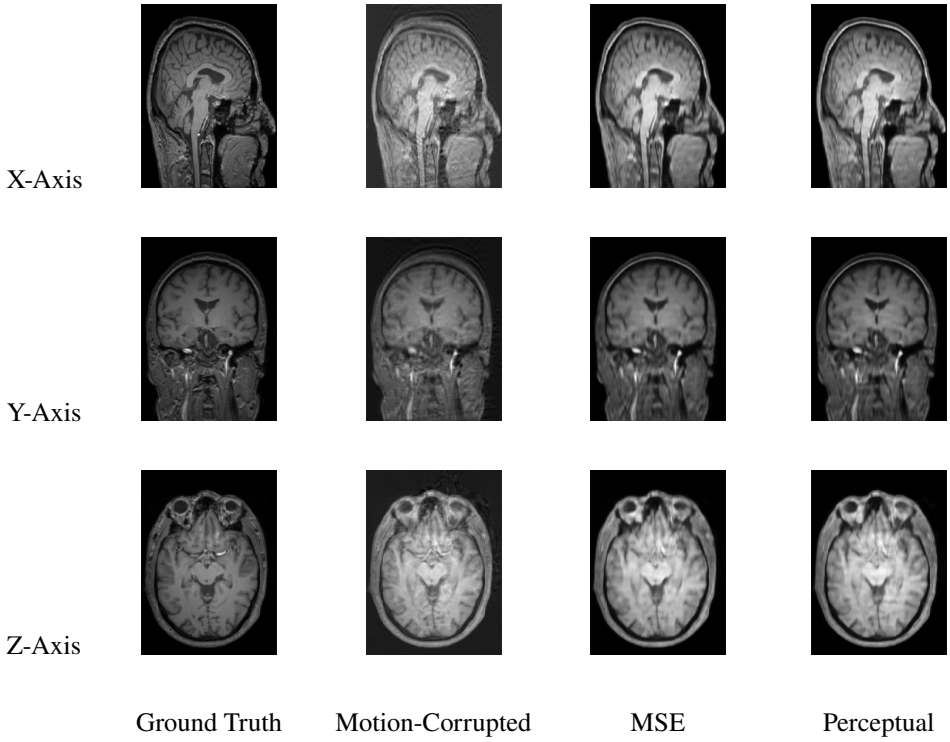

Fig. 10 Motion Correction with dilated convolutions. The sample is taken from the test set.

This is reflected also in the metric results, showed in Table 9, 10, 11 and 12, concerning the training set, and Table 13, 14, 15 and 16, concerning the test set.

Mean Squared Error			
Axis	Corrupted	MSE	Perceptual
x	0.0279	0.00222	0.00212
y	0.0256	0.00186	0.00176
z	0.0165	0.00161	0.00162
Tot.	0.0233	0.00189	0.00184

Table 9 MSE results for the network with dilated convolutions applied on the training set

MAPE			
Axis	Corrupted	MSE	Perceptual
x	15.32%	3.42%	3.25%
y	14.45%	3.37%	3.16%
z	11.27%	3.1%	2.88%
Tot.	13.68%	3.3%	3.09%

Table 10 MAPE results for the network with dilated convolutions applied on the training set

SSIM			
Axis	Corrupted	MSE	Perceptual
x	0.404	0.795	0.806
y	0.435	0.792	0.799
z	0.477	0.829	0.827
Tot.	0.439	0.806	0.811

Table 11 SSIM results for the network with dilated convolutions applied on the training set

SNR			
Axis	Corrupted	MSE	Perceptual
x	15.83	26.87	27.07
y	16.40	27.64	27.87
z	18.73	28.24	28.23
Tot.	16.98	27.58	27.72

Table 12 SNR results for the network with dilated convolutions applied on the training set

Mean Squared Error			
Axis	Corrupted	MSE	Perceptual
x	0.0283	0.00373	0.00395
y	0.0253	0.00332	0.00349
z	0.0231	0.00375	0.00379
Tot.	0.0256	0.00360	0.00375

Table 13 MSE results for the network with dilated convolutions applied on the test set

MAPE			
Axis	Corrupted	MSE	Perceptual
x	15.35%	4.15%	4.10%
y	14.39%	4%	3.83%
z	13.76%	4.16%	3.98%
Tot.	14.5%	4.1%	3.97%

Table 14 MAPE results for the network with dilated convolutions applied on the test set

SSIM			
Axis	Corrupted	MSE	Perceptual
x	0.421	0.746	0.744
y	0.445	0.744	0.733
z	0.479	0.778	0.769
Tot.	0.448	0.756	0.748

Table 15 SSIM results for the network with dilated convolutions applied on the test set

SNR			
Axis	Corrupted	MSE	Perceptual
x	15.73	24.37	24.18
y	16.33	25.16	24.95
z	16.82	24.32	24.34
Tot.	16.29	24.62	24.49

Table 16 SNR results for the network with dilated convolutions applied on the test set

Also in this case, the results obtained using MSE and the Perceptual loss does not differ so much. However, compared to the outcomes achieved in the case without dilated convolutions, the results are quite worse, both from metric and image inspection points of view. To conclude the analysis of the results, the network performs well with both loss functions tested in this work, however images with an higher quality are obtained when Perceptual loss is used. Additionally, the network with dilated convolutions produces images that are more blurred compared to the corrected images obtained without them, so for this kind of task it is discouraged its use.

VIII. Conclusion

In this work i show how, through the use of deep learning algorithms, is possible to recover a clean MR image starting from a motion-corrupted one. The architecture used to accomplish the task is an autoencoder, which uses encoders, that learn to compress the relevant information inside the corrupted images, discarding the corruption, and decoders, that learn to reconstruct a clean image. Two loss functions were tested: Mean Squared Error and Percptual loss, which requires the use of the pre-trained network VGG-16. The differences in the results obtained with them is not that great, however, observing the quality of the images, the corrected slices obtained when using the Perceptual loss are more defined, with details more clear and less blurred. Also dilated convolutions were tested in the Convolutional layers of both encoders and decoders, however the results achieved in this case are worse and not satisfying compared to the previous case. The motion-correction task was accomplished and the algorithm can be concretely used in real world applications. For future applications, the network can be compressed and optimized in order to run on an FPGA accelerator, so to perform motion-correction in real time.

References

[1] Pawar, K., Chen, Z., Jon Shah, N., and Egan, G. F., "MoCoNet: Motion Correction in 3D MPRAGE images using a Convolutional Neural Network approach." *arXiv:1807.10831*, 29 July 2018.

[2] Shaw, R., Sudre, C. H., Varsavsky, T., Ourselin, S., and Cardoso, M. J., "A k-space Model of Movement Artefacts:Application to Segmentation Augmentation and Artefact Removal." *IEEE Transactions on medical imaging, Vol. 39, No. 9*, September 2020.

[3] Duffy, B. A., Zhang, W., Tang, H., Zhao, L., Law, M., Toga, A. W., and Kim, H., "Retrospective correction of motion artifact affected structural MRI images using deep learning of simulated motion." *1st Conference on Medical Imaging with Deep Learning*, 2018.

[4] Usman, M., Umar Farooq, M., Siddique, L., Asim, M., and Qadir, J., "Motion Corrected Multishot MRI Reconstruction Using Generative Networks with Sensitivity Encoding." *arXiv:1902.07430v6 [cs.CV]*, 12 March 2020.

[5] Goodfellow, I., Pouget-Abadie, J., Mirza, M., Xu, B., Warde-Farley, D., Ozair, S., Courville, A., and Bengio, Y., "Generative adversarial nets." *Advances in neural information processing systems, pp. 2672-2680*, 2014.

[6] Ghodrati, V., Shao, J., Bydder, M., Zhou, Z., Yin, W., Nguyen, K., Yang, Y., and Hu, P., "MR image reconstruction using deep learning: evaluation of network structure and loss functions." *Quant Imaging Med Surg 2019;9(9):1516-1527*, 2019.

[7] Alexa, M., "Linear combination of transformations." *ACM Trans. Graph., vol. 21, no. 3, pp. 380-387*, July 2002.

[8] Veit, A., Wilber, M., and Belongie, S., "Residual Networks Behave Like Ensembles of Relatively Shallow Networks." *arXiv:1605.06431v2 [cs.CV]*, 27 October 2016.

[9] Igel, C., and Hüskel, M., "Improving the Rprop Learning Algorithm." *Proceedings of the second international symposium on neural computation*, 2000.