Deep learning techniques for MRI image acquisition and improvement

Virginia Sanfilippo 1652696 and Claudia Melis Tonti 1888489

La Sapienza University of Rome

Abstract. The MRI (meaning Magnetic Resonance Imaging) is a non-invasive medical imaging modality which provides high resolution and appropriate contrast of soft tissues. Despite the advantages, the long-data acquisition time of MRI and the high probability of presence of motion in the resulting images, cause many difficulties in its clinical and research applications. The aim of our project is improving the real-time image acquisition, simulating the motion noise and attempting to fix it with deep learning.

1 Introduction

The motions resulting from the MRI long-data acquisition time can be either rigid or non-rigid. The non-rigid motions, for example the arterial pulsation, are periodic and therefore their effect can be avoided easily. While the rigid motions, such as the head movements that we are interested in, are more difficult to predict and to correct, so, the degradation is more severe.
Consequently, this kind of motion may negatively impact radiological interpretation, for that reason motion correction techniques are essential in MRI reconstruction processes to provide a correct diagnosis.
To achieve the goal of reconstructing static images, we start from a MRI dataset of brains, with some simulated motion added to the set of images through a motion artifact augmentation technique in order to create some corrupted samples of them.
Afterwards, we train a Deep Convolutional Generative Adversarial Network, feeding it both the original and the simulated corrupted images, in order to build a model able to recover the original images starting from the corrupted ones.
In our project we worked with a dataset of *NIfTI* files, format used for neuroimaging, and each file is composed by three slices of MRI.
Figure 1 shows an example of the files in the dataset: the first row shows the original MRIs, while in the second one there are the analogous corrupted images.

1.1 Background and Related Work

MRI acquisition is sensitive to rigid motion, so, the process of reconstruction has the goal of motion correction and it produces an artifact-free image.
To achieve the goal the Retrospective Motion Correction (RMC) is applied to the rigid motion correction process.
The RMC is performed mostly on k-space data acquisition. In particular, Byder

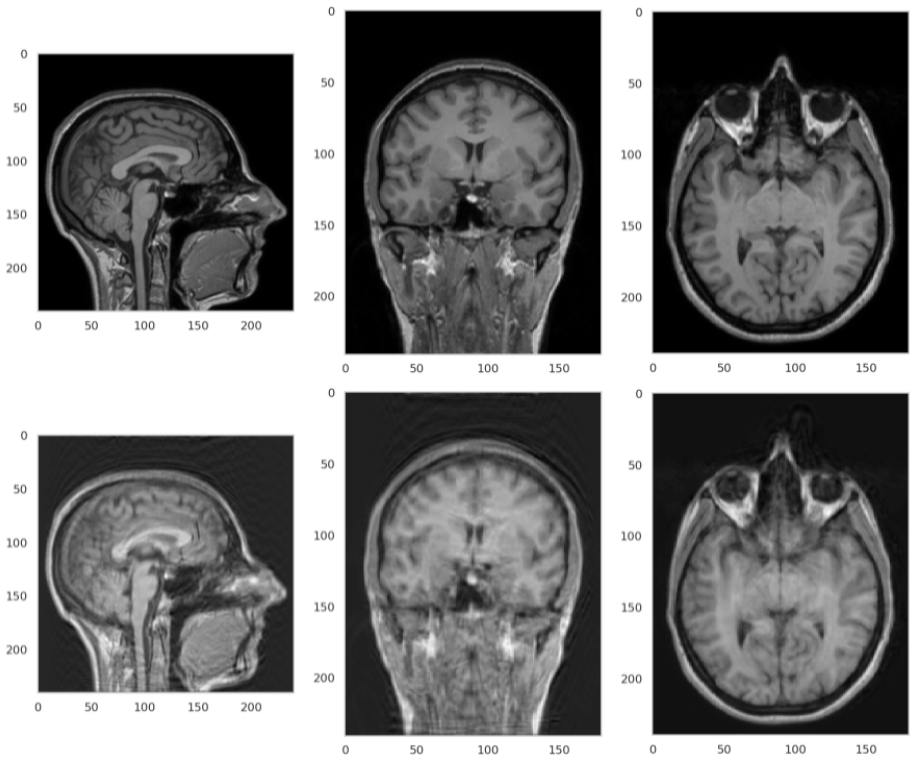

Fig. 1: NIfTI dataset sample image.

et al. studies the k-space inconsistencies by parallel imaging technique. Once the inconsistencies are detected they are discarded and replaced with consistent data. The resulting image has less artifacts and lower signal to noise ratio. In some other works the detection of the motion trajectory is proposed. To achieve this goal, Loktyushin et al. proposes a Gradient-based optimization on the search space exploration. Then, the image is divided into small windows with rigid motions and forwarded to model in order to optimize the unknown motion parameters exploration. The same technique is used by Cordero et al. to correct motion artifacts. The most used models in Deep Learning to correct MRI images are CNNs. However, in [1] the author uses a GAN for motion correction, because it is a type of network highly used for bio-medical image analysis and modeling of natural images. This work is extended with CG-SENSE reconstruction framework for motion correction. The model that we propose is a Deep Convolutional GAN (DCGAN) for the reconstruction of the images corrupted by rigid motions.

1.2 Motion Artifact Simulation

The images in our dataset have undergone corruption process to simulate the human movement that could happen during the acquisition. The corruption is added to the images with a MRI k-Space Motion Artifact Augmentation method: the patient's movement is modeled as a sequence of random 3D affine transformations, which resemble artefact-free volumes, and are afterwards combined in the k-space. The k-space mentioned is an array of numbers representing spatial frequencies in the MR image.

In Figure 2 we have a visual representation of the Motion Artifact Augmentation on MRI scans.

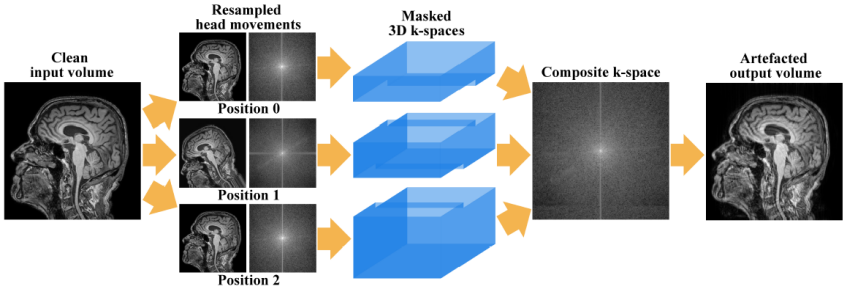

Fig. 2: MRI k-Space Motion Artifact Augmentation.

The dataset has been built by taking a single shot and applying these steps:

1. Starting from a clean input model, a random movement is generated by sampling different probability distribution functions. These movements are rotations from $-10°$ to $10°$ and translations from -10 to 10 mm. These roto-translation transformations are applied with the use of a sequence of matrices A.
2. The generated movement transformations are de-meaned by applying this formula on the average matrix A: since movements at different parts of the k-space contribute different spatial frequencies, we weight each matrix A_i by its signal contribution to the final image.
3. We resample the artefact-free volume according to the de-meaned movement model.
4. A composite k-space is reconstructed from the marked 3D k-spaces of multiple resampled volumes.
5. We transform the k-space to the image domain to produce the final artefacted sample, that will be our output volume.

1.3 Deep Learning To Remove Artifacts

Our goal was to implement a system to correct the motion artefacts, to do so we implemented a DCGAN (Deep Convolutional Generative Adversarial Network)

framework. The model of a Generative Adversarial Network includes a generator G and a discriminator D which play the two-player min-max game:

$min_G max_D E_x[log(D(x))] + E_z[log(1 - G(z))]$.

The Deep Convolutional GAN is a GAN that includes convolutional blocks (convolutional layer, batch normalization layer, leaky RELU layer).

Once trained, our GAN's task, in particular our generator's task, is to recover an original image from the one corrupted in the previous step. Therefore in this paper we use a conditional GAN where, instead of random samples, we use corrupted MRI images as input of the generator.

The final loss of the generator is the sum of the adversarial training loss for G and the data mismatch term.

$L = L_{data} + \lambda L_{adv}$.

With λ being the hyperparameter that controls the weight of each loss term.

2 Implementation

The project has been implemented through Google Colab, using Python and Tensorflow 2.x functions.

2.1 Dataset

The dataset used for the project is composed of 577 MRI images of healthy patients, where each MRI is composed by multiple slices, in order to allow us to analyze the severity of the rigid motions. The dataset contains both the original static MRI and the MRI corrupted with motion artifacts.

The motion artifacts have been added to the images after a conversion into the k-space, using a reconstruction algorithm, where they have been corrupted, and then they have been reconstructed back to the original space.

2.1.1 NIfTI: Neuroimaging Informatics Technology Initiative

The dataset is composed of NIfTI images, a specific format created to have a targeted service in the Neurological field, used to store brain imaging data. This data format allowed us to see three slices of MRI scan, so three different perspectives, for each patient, like we can see in Figure 1.

2.1.2 Image Preprocessing

The NIfTI format made the processing of the images more complex through a model, since, having three slices per file, it required to operate in a 3D manner, with all the difficulties of the sort, especially when we consider the case with coherent and proportionate resulting images. To avoid such problems, we split the images into the three perspectives, processing one by one scan x, y and z. Doing so, we could apply a scaling that worked for all three axes, without

converting the files into jpeg or png format, as it would have caused a consistent image quality loss. We used the NiBabel library package to read, normalize, resize and process the scans.

2.1.3 CG-SENSE

A popular method for motion artifact could have been the CG-SENSE algorithm. This algorithm utilises a conjugate gradient to efficiently solve the SENSE equations. These equations, sensitivity encoding, are based on the fact that the receiver sensitivity has an encoding effect complementary to Fourier preparation by linear field gradients.
The SENSE equation relates gradient encoding, sensitivities and aliased images to unaliased ones, reconstructing the latter in the image domain. CG-SENSE algorithm relates the object to be imaged x_m, the encoding matrix E and the acquired k-space data y:

$Ex_m = y.$

The CG algorithm iteratively solves this equation. Using the Moore-Penrose inversion we can easily compute Z, that is the reconstruction matrix.

$Z = (E^H E)^{-1} E^H.$

For our project we didn't use CG-SENSE to implement image corruption, but only an augmented k-space based corruption.

2.2 GAN

For our model, firstly we tried implementing a Convolutional GAN, which did not give optimal results, because of the presence of the *Conv2DTranspose* layers inside the generator's decoder section. The *Conv2DTranspose* layers' role is to produce an upsampling without loss of information, but the results showed how the generator "tricked" the discriminator by lightening the images. To avoid this, we decided to replace these layers with simple *Upsampling* ones: the image quality is reduced, but the corruption is cleared up. At first we used the bilinear method for Upsampling, but the results seemed that the images were sectioned by a sort of grid. So, we then used the *nearest neighbors* method for the Upsampling. The details of the resulting images were smoother, but the blur was higher with respect to the previous ones. Plus, although the bilinear method gives "grid" images, it maintains more details. So, to have a middle ground result, we used the bilinear method for the first layer of the generator's encoder and then the other layers are upsampled with nearest neighbors method.

2.2.1 Structure

The GAN (Generative Adversarial Network) we built consists of a generator and a discriminator which play the two-player min-max game.

The generator (Figure 5) is constructed as a functional model. It is consisted by an encoder which includes a sequence of *Conv2D* layers blocks, composed by a *Conv2D* layer followed by a *LeakyReLU* activation function layer and by batch

normalization layer, and a decoder which includes the aforementioned *UpSampling2D* layers, also followed by *LeakyReLU* layer and by batch normalization layer.

The discriminator (Figure 6) is similarly constructed as a functional model, using Keras functions and layers. We now consider two inputs, which will be the corrupted and to the corresponding original image while training. The model consists in a sequence of *Conv2D* layers blocks, followed by a *Dropout* layer.

2.2.2 Training Process

To train our model we developed a system that gives to the GAN as input both the original images and the corresponding corrupted ones. We took pairs of original/corrupted images, shuffled and normalized them and then passed to the model as a pair.

We used the **Wasserstein Loss**, which uses the Wasserstein Distance to compute the loss. The Wasserstein distance is computed as follows:

$W := W(F_A, F_B) = (\int_0^1 |F_A^{-1}(u) - F_B^{-1}(u)|^2 \, du)^{1/2}$.

Where F_A and F_B are the cumulative distribution functions of the two conditions A and B

The considered distance is the squared Wasserstein distance ($d := W^2$):

$d := d(F_A, F_B) = \int_0^1 |F_A^{-1}(u) - F_B^{-1}|^2 \, du = (\mu_A - \mu_B)^2 + (\sigma_A - \sigma_B)^2 + 2\sigma_A\sigma_b(1 - \rho^{A,B})$.

We also performed various ablation tests, meaning we removed various elements in the model, in order to verify the effect on the results. First of all, we tried to train the model without Leaky ReLu layers, then we tested the model without Batch Normalization Layers.

3 Results

In Figure 3 we can see the results of the validation test on one corrupted image through the various GANs developed: we can see how the models that use *Conv2DTranspose* layers generate a visibly lighter images, while the final model, that uses *UpSampling2D* layers only, generates a result which contrast is higher and is more similar to the ground truth.

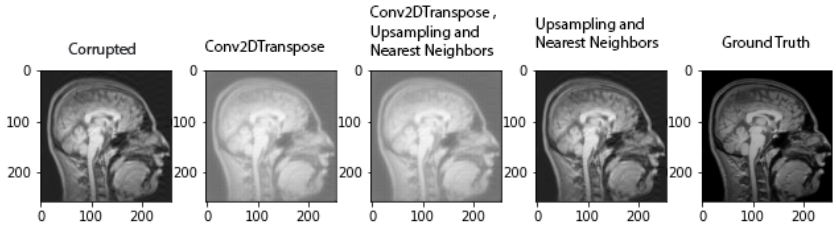

Fig. 3: Results comparison between the corrupted image, the various GAN generated ones and the ground truth.

3.0.1 Loss and Ablation Tests

To analyze the DCGAN training process we used the Binary Cross Entropy loss to compute the discriminator loss when it compares the corrupted image with the original one and the generated image with the original one. The total discriminator loss is the sum of the two computed losses. For the loss of the generator we sum the l1 loss multiplied by a constant λ (that we set 0.5) summed by the GAN loss. This last one is computed with the Wasserstein distance.

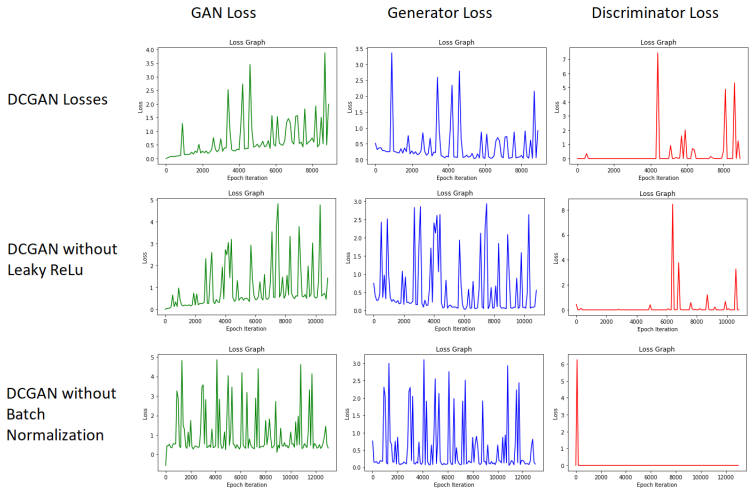

Fig. 4: Loss graphs comparison between models of the Ablation Test

The Figure 4 compares the losses of the models of the ablation test. The plots took data from the training process along 10 epochs.
The first row shows the losses of our final DCGAN with presence of Leaky ReLu and Batch Normalization for each convolutional block of the generator. We can see that despite the total Wasserstein loss of the DCGAN is increasing, the loss of the generator is decreasing along the epochs. This means that the GAN doesn't lack of the possibility of achieving a convergence.
On the second row we analyze the model without Leaky ReLu layers. We can see that the performances are very similar to the ones of the previous model. On the generator we can notice that there are more peaks.
Finally, in the third row we analyze the model without Batch Normalization. We can notice that the GAN loss suddenly increase and then is constant along the epochs, while the discriminator has a peak at the start and then is constant and close to 0. These two graphs may be symptoms of a Convergence Failure of the GAN, as we can learn from [2].

4 Conclusions

In conclusion, we introduced a flexible method that can satisfy the need for the removal of the motion artifacts in multishot Magnetic Resonance Imaging (MRI), in order to help diagnosis. This problem can be approached through Deep Learning, specifically through a Generative Adversarial Network. The technique explored firstly adds a motion corruption in k-space data to the MRI scans, then the resulting images are passed to our GAN framework, that removes the motion and attempts to restore the original image. We performed several experiments to validate our methods, varying different parameters and interchanging the use of different layers and activation functions. A future extension in this field could be improving the resolution of the generated images, possibly through the concatenation of multiple models.

List of Figures

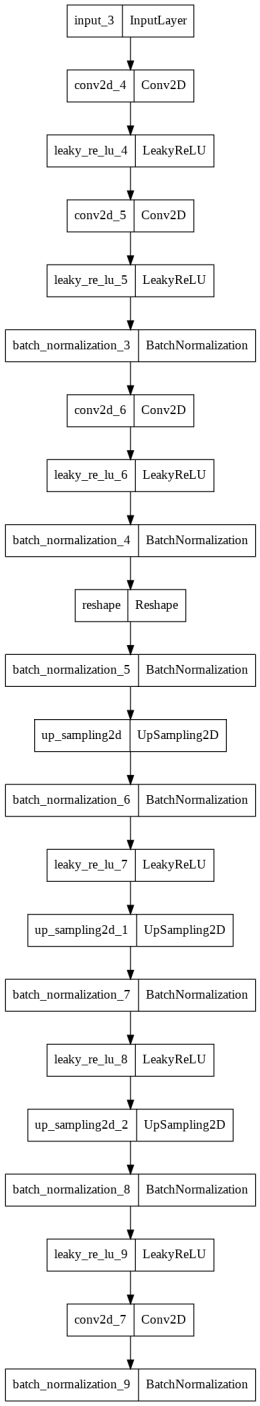

Fig. 5: GAN: Generator's structure.

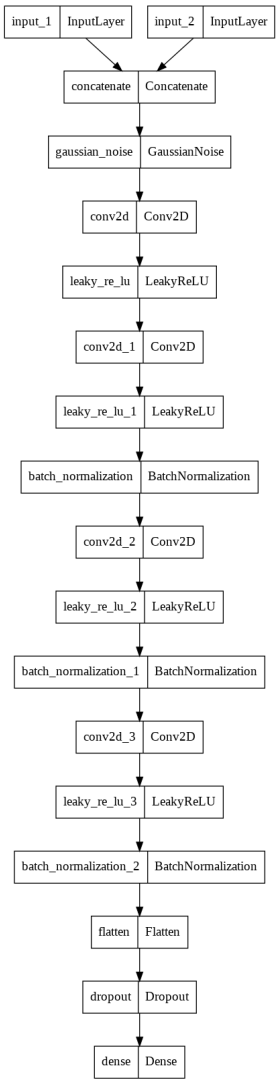

Fig. 6: GAN: Discriminator's structure.

References

[1] Siddique Latif Muhammad Asim Muhammad Usman, Muhammad Umar Farooq and Junaid Qadir. Motion corrected multishot mri reconstruction using generative networks with sensitivity encoding, 2020.

[2] Jason Brownlee. How to identify and diagnose gan failure modes. https://machinelearningmastery.com/practical-guide-to-gan-failure-modes/, 2019.

[3] Sebastien Ourselin M. Jorge Cardoso Richard Shaw, Carole Sudre. Mri k-space motion artefact augmentation: Model robustness and task-specific uncertainty, 2019.

[4] Alexander Fyrdahl Kerstin Hammernik Seb Harrevelt Lars Kasper Agah Karakuzu Michael Loecher Franz Patzig Ye Tian Ke Wang Daniel Gallichan Martin Uecker Florian Oliver Maier, Steven H. Baete. Cg-sense revisited: Results from the first ismrm reproducibility challenge, 2020.

[5] Haoteng Tang Lu Zhao Meng Law ArthurW. Toga Ben A. Duffy, Wenlu Zhang and Hosung Kim. Retrospective correction of motion artifact affected structural mri images using deep learning of simulated motion, 2018.