## Introduction

## Magnetic resonance imaging (MRI) is a medical imaging technique used in radiology to form pictures of the anatomy and the physiological processes of the body. MRI scanners use strong magnetic fields, magnetic field gradients, and radio waves to generate images of the organs in the body[[1]](#footnote-1). Although MRI is a powerful tool for diagnosis, it can introduce motion artifacts in the image resulted by the slightest movement of the patients during scanning. Such artifacts may result in less accurate images and can impede the work of professionals during diagnosis.

Therefore, the aim of this project is to classify MRI images into two categories: those which present motion artifacts called “motion” and those where the patients stood still during scanning, “nomotion”, and potentially serve as a tool for professionals during the analysis of such images.

## Data Acquisition

The first step was the data acquisition[[2]](#footnote-2). After downloading the .nifti files, it was necessary to check that the data is correctly labeled, and for unlabeled data it had to be decided what “highres-run01” and “highres-run02” represented. There were inconsistencies in the files denoted with “highres”, namely: almost all the run02 files are in motion and run01 are not in motion. The only exceptions are for the patients: NC216 and NC245 where run01 -> motion, run02 ->not motion. Once the labels were correctly matched, the files were renamed with the code of the patient followed by the label: motion or nomotion.

## Preprocessing:

The images are 3-dimensional and so all the points are at the intersection of three planes. For the project it was decided to work only on the sagittal plane, each sagittal slice having the dimension 256x256.

The first preprocessing task was to apply the N4 Bias Field Correction[[3]](#footnote-3) which corrects low frequency intensity non-uniformity present in MRI image data known as a bias or gain field. The transformation could not be visible by the naked eye but subtracting the image arrays showed that the difference was significant.

The second step was to scale the pixels to the 0-255 interval (the pixels being integers). To do this a linear function was used that maps the minimum and the maximum of an image to 0 and 255 respectively.

For the last preprocessing step, Contrast Limited Adaptive Histogram Equalization[[4]](#footnote-4) (CLAHE) was applied on every sagittal slice. CLAHE is a variant of histogram equalization which amplifies the contrast in the image but also takes care of over-amplification, unlike Adaptive Histogram Equalization (AHE).

After applying CLAHE, each slice of the .nifti files was saved on the disk with an appropriate name that contains the code of the patient, the number of the slice and the associated label.

## Dataset splitting:

First and foremost, on the disk each image represents a slice of a .nifti file. It is crucial to mention that the data is perfectly balanced as exactly 50% of the slices are in motion and the other 50% are not in motion.

During train-validation-test splitting all the slices of a single patient, regardless of their label, go to a single partition to insure against data leakage. The train-validation-test default proportion is 80%-15%-5%.

## Convolutional Neural Networks (CNNs)

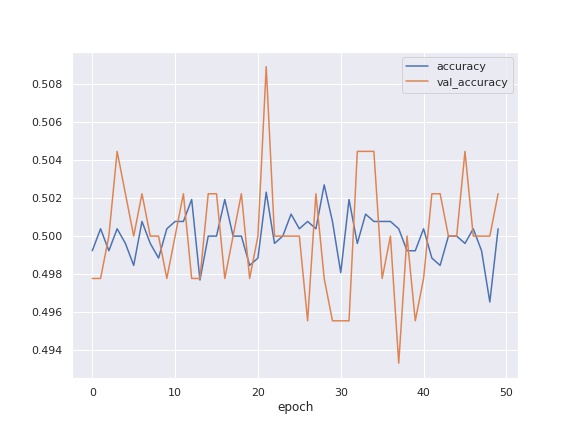
This is a list of things that were noticed while trying out different models.

As previously mentioned, the task is to classify whether the patient moved during the MRI scan or not. The images with movement have more noise than the ones where the head of the patient stays fixed. However, that noise is an attribute of the whole image, and it does not appear just in some isolated places.

During the attempts, the following architecture of a blur detection model suggested by the literature was considered:

* BDNet[[5]](#footnote-5), which was conceived for blur detection. The attempt was not fruitful as the images it has to classify are very different from the ones where BDNet might excel. BDNet is supposed to detect a region where the camera is out of focus. Nevertheless, the MRI images are always in focus whether they present motion artifacts or not. Moreover, images do not contain just a small portion that moves, rather the whole image “shakes” which results in a noise in the form of a trace of translations and rotations[[6]](#footnote-6).

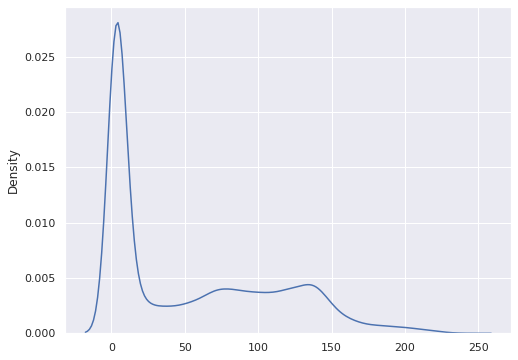
The following image corresponds to the model described above and it shows that the model does the same job overall as a classifier that would assign one class to all the images:



* Also, in literature it is suggested to use at first big kernel sizes in order to capture as many features as possible and to also choose a great number of filters. During the attempts it was observed that a combination of 5x5 and 3x3 filters work better than one or even multiple 7x7 filters. It is important to note, however, that the effective receptive field resulted by stacking up a 5x5 and a 3x3 kernel is the same as a 7x7 kernel[[7]](#footnote-7). This may indicate that motion artifacts tend to keep close to the object from which they resulted. Moreover, the more filters a layer applies the harder it seems for the model to work out the difference between images.
* It is generally recommended to use MaxPooling layers in favour of AveragePooling layers[[8]](#footnote-8). However, after a visual inspection of the images it was concluded that images with motion tend to have more light pixels (that is higher pixel values) due to the noise. This way, the average pixel value in a region of interest will be higher for motion images. But a MaxPooling layer cannot always capture this phenomenon as if two images labeled differently both have lighter regions, then the maximum pixel value will be similar for both. It remains to be seen if a MedianPooling would behave better as the distribution of the noise has not been studied but it is unlikely to result in a significantly higher accuracy.
* Moreover, although in the last few years BatchNormalization[[9]](#footnote-9) layers have shown significant improvements on CNNs, in this project they bear a negative effect on accuracy. This is because BatchNormalization normalizes the distribution of the pixels in an image. However, having a distribution that is not normal is key to the classification as the distribution of an image with motion differs from the distribution of an image with no motion. The following plots represent the kernel density estimation of pixels for a “motion” and a “nomotion” image. Both images represent the same slice number from the same patient, but one from the scanning with motion and one from the scanning without motion. Although highly similar, the difference can be spotted even without differencing the two plots.

“motion” image “nomotion” image

Chart, line chart, histogram

Description automatically generated

The plots of the training for different CNN models, which are all included in the model\_report.pptx powerpoint, exhibit a rather odd behavior. On average, a baseline model should have constantly an accuracy around 0.5 as the classes are perfectly balanced. However, the training of multiple CNN models starts with an accuracy of 0 and many times not even reaching 0.5 after more than 50 epochs. The following plot represents the evolution of accuracy during 50 epochs, and it shows that the accuracy on validation plummets after the first epoch and does increase significantly after that.

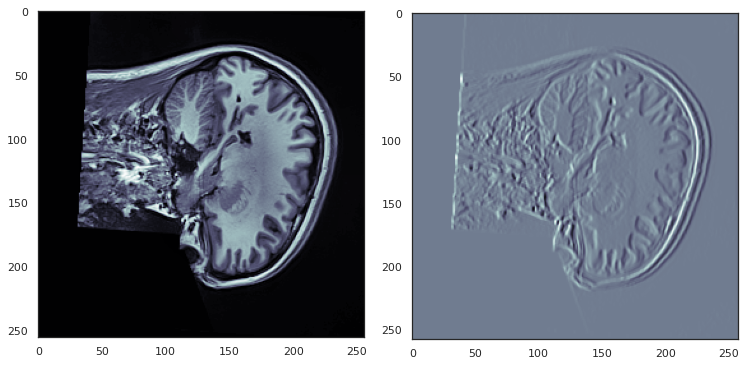


Such models on testing predicted all images as “nomotion”. Therefore, in this particular project, it seems the CNNs tend to discriminate “motion” images which can indicate that motion artifacts fade after multiple layers of convolution and kernel operations.

To prove this assumption, a Laplacian kernel for edge detection[[10]](#footnote-10) was applied on a “motion” and a “nomotion” image.

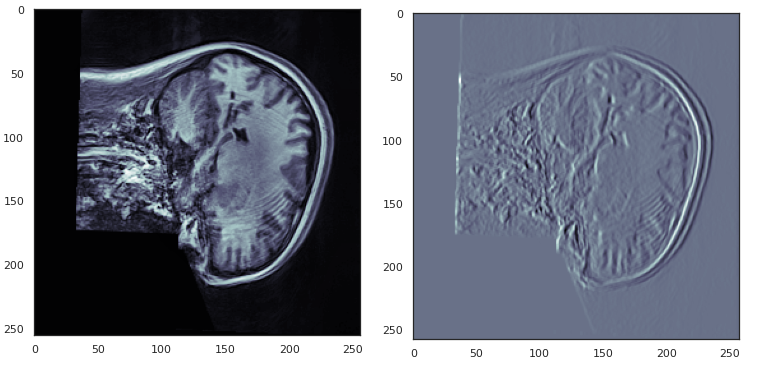
* nomotion

original transformed



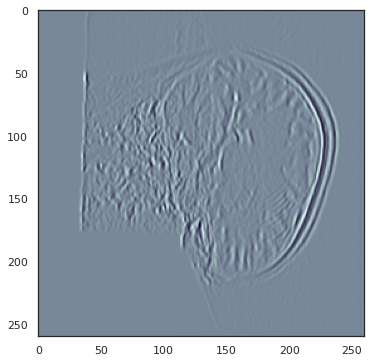
* motion

original transformed



Applying the transformation again it was noticed that the motion artifacts begin to fade and so the only differentiating characteristic of the motion images disappears by applying a filter multiple times.

Laplacian applied twice on a motion image



Next it was necessary to see how the image would look like after passing through the layers of a convolutional model. Thus a simple model with only one convolutional layer with 1 filter and a 3x3 kernel was considered and after choosing a suitable checkpoint, the weights of the layer were applied on a sample of images. On each row the resulting kernel is applied to the image and we can see the difference between a “motion” and a “nomotion” image. Although visually we can still see the motion artifacts (albeit more faded), the difference between the two classes diminishes considerably. To be noted that this is a simple model that applies only one filter, hence a more sophisticated network would further fade away the differences between the two images. Testing has shown that it reduces the difference to the point where all images are classified as “nomotion”.

motion nomotion

Graphical user interface

Description automatically generatedGraphical user interface

Description automatically generated

Graphical user interface

Description automatically generatedA picture containing graphical user interface

Description automatically generated

All the remarks suggest that CNNs may not be the best approach to dealing with the task of this project, thus a different approach was chosen which lead to unexpected results.

## Back to the basics

Considering the unsuccessful trials with CNNs the next step was to try to construct a model as basic as it gets and try to gradually increase the accuracy without running the risk of removing important details from the images.

As a stroke of luck which more experienced people would call “intuition”, the first such trial was the problem solver of this whole project. Dense layers were considered with a varied the number of units, a layer to flatten the tensor and a final Dense layer of 1 unit for prediction.

The activation function chosen for the first layers was RELU and for the prediction layer sigmoid.

The next models are evaluated using stratified cross validation evaluation[[11]](#footnote-11) which introduces variation in the data used for fitting the model.

## Stratified cross validation

For stratified cross validation, the images are shuffled and split in 5 folds. Stratified means that each fold has the same proportion of motion/nomotion images (in our case each fold has exactly half of the data as “motion” and the other half as “nomotion”). Before splitting in folds, a proportion of the data is reserved for testing, 5% is the default. One fold will be used for validation and four for training. The model will be fit from scratch 5 times, each time the validation fold will be changed (so that the model is validated each time on different data) and the remaining folds will be used for training.

Each time the model will be evaluated on the data reserved for testing and the script will report the values resulted for specificity, recall, but also a plot with the average accuracy per epoch with the interval of confidence.

This mechanism of evaluation introduces variety in the data used for training and validation and rules out the chance that the model performs well only on a well-behaved data set. The only issue with the cross validation is that the seed for keras was not set, and thus randomness plays a role in the evaluation of the models. Although on this evaluation the “Best model” does not have the highest metrics, on repeated runs of the model it was observed that it is more consistent with the evaluation compared to the other ones.

## Models with dense layers

Best model: Dense with 64 units

Layers:

* Dense: 64 units, activation = relu
* Flatten
* Dense: 1 unit, activation = sigmoid

**Cross validation evaluation:**

**Specificity scores = [0.66, 0.77, 0.67, 0.65, 0.75]**

**Specificity mean = 0.70**

**Recall scores = [0.5, 0.71, 0.51, 0.46, 0.75]**

**Recall mean = 0.58**

Chart

Description automatically generated

### Dense with 48 units

Layers:

* Dense: 48 units, activation = relu
* Flatten
* Dense: 1 unit, activation = sigmoid

**Cross validation evaluation:**

**Specificity scores = [0.55, 0.8, 0.76, 0.68, 0.80]**

**Specificity mean = 0.72**

**Recall scores = [0.19, 0.75, 0.69, 0.53, 0.76]**

**Recall mean = 0.58**

Chart, line chart

Description automatically generated

### Dense with 36 layers

Layers:

* Dense: 36 units, activation = relu
* Flatten
* Dense: 1 unit, activation = sigmoid

**Cross validation evaluation:**

**Specificity scores = [0.53, 0.8, 0.62, 0.65, 0.72]**

**Specificity mean = 0.66**

**Recall scores = [0.13, 0.75, 0.40, 0.48, 0.69]**

**Recall mean = 0.49**

Chart, line chart

Description automatically generated

### Dense with 16 units

Layers:

* Dense: 16 units, activation = relu
* Flatten
* Dense: 1 unit, activation = sigmoid

**Cross validation evaluation:**

**Specificity scores = [0.76, 0.81, 0.59, 0.72, 0.77]**

**Specificity mean = 0.73**

**Recall scores = [0.71, 0.76, 0.30, 0.67, 0.75]**

**Recall mean = 0.64**

Chart

Description automatically generated

### Dense with Dropout

Layers:

* Dense: 64 units, activation = relu
* Dropout: 0.3
* Flatten
* Dense: 1 unit, activation = sigmoid

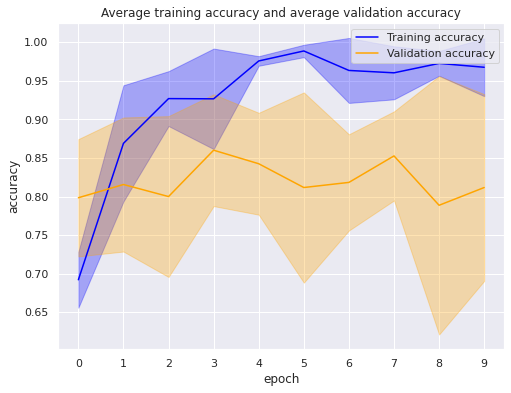
**Cross validation evaluation:**

**Specificity scores = [0.65, 0.72, 0.72, 0.67, 0.77]**

**Specificity mean = 0.71**

**Recall scores = [0.48, 0.75, 0.61, 0.51, 0.76]**

**Recall mean = 0.62**



### Two dense layers

Layers:

* Dense: 64 units, activation = relu
* Dense: 64 units, activation = relu
* Flatten
* Dense: 1 unit, activation = sigmoid

**Cross validation evaluation:**

**Specificity scores = [0.67, 0.72, 0.72, 0.72, 0.81]**

**Specificity mean = 0.73**

**Recall scores = [0.51, 0.61, 0.61, 0.61, 0.76]**

**Recall mean = 0.62**

Chart, line chart

Description automatically generated

### Two dense layers, reduced units on the second layer

Layers:

* Dense: 64 units, activation = relu
* Dense: 32 units, activation = relu
* Flatten
* Dense: 1 unit, activation = sigmoid

**Cross validation evaluation:**

**Specificity scores = [0.72, 0.73, 0.72, 0.66, 0.76]**

**Specificity mean = 0.72**

**Recall scores = [0.61, 0.76, 0.61, 0.55, 0.76]**

**Recall mean = 0.66**

Chart

Description automatically generated

## Data augmentation

In the last part of the project data augmentation was tried with the hope of aiding convolutional neural networks, but also dense networks, to learn relevant features and rule out the possibility to have them predict based on irrelevant information, such as the position of the brain in the image.

### Introducing translations on the training images

Overall, the results of introducing translations were slightly worse.

Model: Dense with 64 units

Layers:

* Dense: 64 units, activation = relu
* Flatten
* Dense: 1 unit, activation = sigmoid

**Cross validation evaluation:**

**Specificity scores = [0.77, 0.71, 0.60, 0.64, 0.73]**

**Specificity mean = 0.69**

**Recall scores = [0.75, 0.71, 0.38, 0.53, 0.71]**

**Recall mean = 0.61**

Chart, line chart

Description automatically generated

#### Convolutional model:

Layers:

* Conv2D: 16 filters, 5x5 size, activation = relu
* Flatten
* Dense: 1 unit, activation = sigmoid

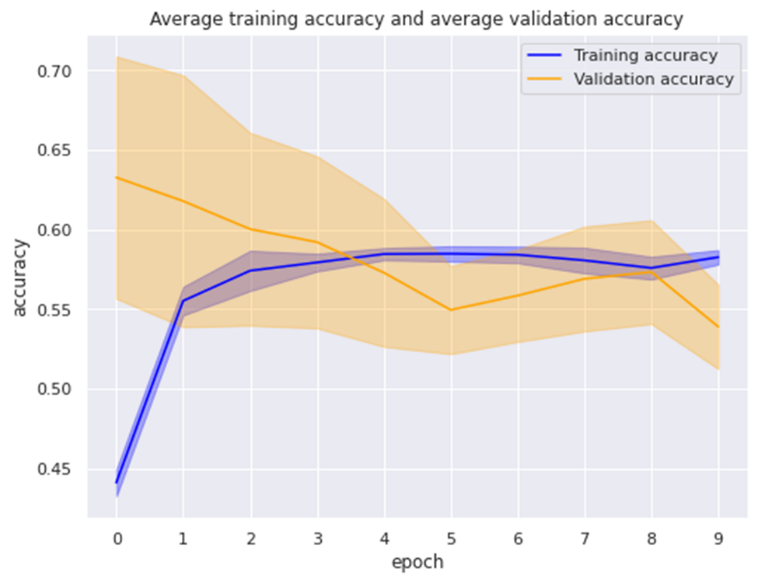
**Cross validation evaluation:**

**Specificity scores = [0.61, 0.56, 0.36, 0.53, 0.53]**

**Specificity mean = 0.52**

**Recall scores = [0.63, 0.57, 0.76, 0.5, 0.36]**

**Recall mean = 0.56**



### Horizontal flips

Horizontal flips alone do not seem to help the model.

Model: Dense with 64 units

Layers:

* Dense: 64 units, activation = relu
* Flatten
* Dense: 1 unit, activation = sigmoid

**Cross validation evaluation:**

**Specificity scores = [0.61, 0.69, 0.65, 0.59, 0.60]**

**Specificity mean = 0.63**

**Recall scores = [0.38, 0.67, 0.71, 0.48, 0.34]**

**Recall mean = 0.51**

Chart, line chart

Description automatically generated

Model: Dense with 64 units and Dense with 32 units

Layers:

* Dense: 64 units, activation = relu
* Dense: 32 units, activation = relu
* Flatten
* Dense: 1 unit, activation = sigmoid

**Cross validation evaluation:**

**Specificity scores = [0.75, 0.76, 0.66, 0.55, 0.70]**

**Specificity mean = 0.68**

**Recall scores = [0.67, 0.76, 0.5192307692307693, 0.63, 0.59]**

**Recall mean = 0.63**

Chart

Description automatically generated

### Rotations

As opposed to the other transformations, rotations seem to bear positive results.

Model: Dense with 64 units

Layers:

* Dense: 64 units, activation = relu
* Flatten
* Dense: 1 unit, activation = sigmoid

**Cross validation evaluation:**

**Specificity scores = [0.60, nan, 0.63, 0.62, 0.66]**

**Specificity mean = nan**

**Recall scores = [0.34, 1.0, 0.48, 0.5, 0.53]**

**Recall mean = 0.57**

Chart, line chart

Description automatically generated

Convolutional model: Conv2D, 8 filters 5x5

Layers:

* Conv2D: 8 filters, 5x5, activation = relu
* Flatten
* Dense: 1 unit, activation = sigmoid

**Cross validation evaluation:**

**Specificity scores = [0.66, 0.59, 0.6, 0.57, 0.75]**

**Specificity mean = 0.63**

**Recall scores = [0.73, 0.55, 0.69, 0.42, 0.73]**

**Recall mean = 0.62**

Chart, line chart

Description automatically generated

Convolutional models

Layers:

* Conv2D: 16 filters, 5x5, activation = relu
* Conv2D: 2 filters, 3x3, activation = relu
* Flatten
* Dense: 1 unit, activation = sigmoid

**Cross validation evaluation:**

**Specificity scores = [0.45, 0.54, 0.31, 0.47, 0.48]**

**Specificity mean = 0.45**

**Recall scores = [0.46, 0.53, 0.78, 0.30, 0.65]**

**Recall mean = 0.55**

Chart

Description automatically generated

### Rotations and translations

Convolutional model

Layers:

* Conv2D: 16 filters, 5x5, activation = relu
* Flatten
* Dense: 1 unit, activation = sigmoid

**Cross validation evaluation:**

**Specificity scores = [0.63, 0.64, 0.54, 0.64, 0.58]**

**Specificity mean = 0.61**

**Recall scores = [0.59, 0.67, 0.80, 0.53, 0.44]**

**Recall mean = 0.61**

Chart, line chart

Description automatically generated

### Scalations

Convolutional models

Layers:

* Conv2D: 16 filters, 5x5, activation = relu
* Conv2D: 2 filters, 3x3, activation = relu
* Flatten
* Dense: 1 unit, activation = sigmoid

**Cross validation evaluation:**

**Specificity scores = [0.43, 0.44, 0.58, 0.44, 0.47]**

**Specificity mean = 0.47**

**Recall scores = [0.44, 0.53, 0.86, 0.63, 0.26]**

**Recall mean = 0.55**

Chart

Description automatically generated

Layers:

* Conv2D: 16 filters, 5x5, activation = relu
* Flatten
* Dense: 1 unit, activation = sigmoid

**Cross validation evaluation:**

**Specificity scores = [0.59, 0.57, 0.43, 0.62, 0.60]**

**Specificity mean = 0.56**

**Recall scores = [0.51, 0.53, 0.82, 0.67, 0.38]**

**Recall mean = 0.58**

Chart, line chart

Description automatically generated

Model: Dense with 64 units

Layers:

* Dense: 64 units, activation = relu
* Flatten
* Dense: 1 unit, activation = sigmoid

**Cross validation evaluation:**

**Specificity scores = [0.50, 0.64, 0.58, 0.62, 0.57]**

**Specificity mean = 0.58**

**Recall scores = [0.03, 0.61, 0.40, 0.44, 0.26]**

**Recall mean = 0.35**

**Chart, line chart

Description automatically generated**

## Data augmentation on BDNet

As a last trial, data augmentation (translations, rotations and scalations) was performed while training the BDNet architecture. The results are not satisfactory as the model predicts all the images as “nomotion”.

**Specificity scores = [nan, nan, nan, nan, nan]**

**Specificity mean = nan**

**Recall scores = [1.0, 1.0, 1.0, 1.0, 1.0]**

**Recall mean = 1.0**

**Chart, line chart

Description automatically generated**

To sum up on the data augmentation, although it is supposed to be aiding the model against overfitting, on this problem and with this dataset it makes the accuracy of prediction worse. This was to be expected as the testing set is extracted from the same set of images as the training and validation sets and the images are consistent with the positioning and orientation of the brain. Therefore, data augmentation introduces variety that is not present on the data used for evaluation. On top of that, the problems resulted by using CNNs are not solved by augmenting the data as the model is still unable to locate the motion artifacts and they fade as the number of filters increases. What is not clear is whether a model trained without data augmentation is as powerful on data from the real world as it is on the data used on this project.

## Conclusion

Although the initial aim of this project was fulfilled, namely finding a model that can classify MRI images into “motion” and “nomotion”, the project is far from being over. The reason is that it is unclear how to separate the motion artifacts in the images with motion.

Moreover, the model is decent at detecting motion in MRI images of healthy patients, but it is unclear how the model would perform on images that present tumors or other injuries.

More importantly, detecting motion is one thing, but it is unclear how the model would serve a real purpose in aiding professionals in diagnosis. It would be ambitious to try and detect the motion artifacts and to artificially remove the motion in the image without altering the image too much.

Finally, although the solution to this model was simpler than expected, it can be considered a lesson in the way such a project should be approached. Too early sophisticated networks suggested on the internet were tried instead of attempting to apply some basic textbook knowledge and go from there. I personally feel I could have done better and commit myself more to this project, but all the failures I had thought me invaluable lessons which would undoubtedly aid me in my further work.

1. Wikipedia: [Magnetic resonance imaging](https://en.wikipedia.org/wiki/Magnetic_resonance_imaging) [↑](#footnote-ref-1)
2. Database: [Openneuro.org](https://openneuro.org/datasets/ds003639/versions/1.0.0) [↑](#footnote-ref-2)
3. SIMPLEITK: [N4 Bias Field Correction](https://simpleitk.readthedocs.io/en/master/link_N4BiasFieldCorrection_docs.html) [↑](#footnote-ref-3)
4. OpenCV: [CLAHE](https://www.geeksforgeeks.org/clahe-histogram-eqalization-opencv/) [↑](#footnote-ref-4)
5. [Blur Detection Convolutional Neural Network](https://sh-tsang.medium.com/review-bdnet-blur-detection-convolutional-neural-network-blur-detection-9150bedd0546) [↑](#footnote-ref-5)
6. Further details: [NIH](https://www.ncbi.nlm.nih.gov/pmc/articles/PMC5447676/#:~:text=Movement%20artifacts%20in%20MRI%20degrade,%2C%20e.g.%2C%20ghosting%20or%20blurring.) [↑](#footnote-ref-6)
7. Salman Khan et al., *A Guide to Convolutional Neural Networks for Computer Vision*, p. 50. [↑](#footnote-ref-7)
8. [Pooling Layers](https://androidkt.com/explain-pooling-layers-max-pooling-average-pooling-global-average-pooling-and-global-max-pooling/) [↑](#footnote-ref-8)
9. [A Gentle Introduction to Batch Normalization for Deep Neural Networks](https://machinelearningmastery.com/batch-normalization-for-training-of-deep-neural-networks/) [↑](#footnote-ref-9)
10. [What is Edge Detection – An Introduction](https://www.mygreatlearning.com/blog/introduction-to-edge-detection/#:~:text=Edge%20detection%20is%20a%20technique,or%20boundaries) [↑](#footnote-ref-10)
11. [What is Stratified Cross-Validation in Machine Learning](https://towardsdatascience.com/what-is-stratified-cross-validation-in-machine-learning-8844f3e7ae8e#:~:text=Implementing%20the%20concept%20of%20stratified,close%20approximation%20of%20generalization%20error.) [↑](#footnote-ref-11)