Institute of Imaging and Computer Vision | RWTH AACHEN UNIVERSITY

A systematic evaluation of diffusion model-based domain transfer - a clinical validation study

Systematische Evaluation des Domänen-Transfers mit Diffusionsmodellen anhand einer klinischen Anwendungsstudie

Bachelor Thesis

presented by

Cederic Aßmann

Supervisor:
Gustav Müller-Franzes, M.Sc.

Institute of Imaging & Computer Vision
Dr.-Ing. habil. Mathias Wien
RWTH Aachen University

Erklärung nach §18 Abs. 1 ÜPO

Hiermit versichere ich, dass ich die vorgelegte Bachelor Thesis selbständig angefertigt habe. Es sind keine anderen als die angegebenen Quellen und Hilfsmittel benutzt worden. Zitate wurden kenntlich gemacht.

I hereby confirm that I have written this Bachelor Thesis independently using no sources or aids other than those indicated. I have appropriately declared all citations.

Cederic Aßmann
Aachen, 01.09.2023

Abstract

The relevance in clinical practice for the transfer of different MRI sequences is occasionally since the time the patient spends in the scanner can be reduced. The aim of this bachelor thesis is the evaluation of diffusion model based domain transfer in MRI-Imaging. The question is focused on whether the model can perform the sequence conversion regarding to the signal properties and regarding to maintain the anatomical structures between the real and generated images. A Diffusion Autoencoder architecture combined with an implemented Edge Optimization algorithm is used, it is adapted to the requirements of medical image analysis methods and trained with MRI sequences of the lower limbs from different datasets such as MRNet and FastMRI. The conversion works better in terms of signal correspondence than anatomical correspondence according to the radiological evaluation. Moreover, the conversion does not work acceptably for each sequence conversion and the metric evaluation differs from the radiological evaluation. In conclusion, the MRI sequence conversion is partially successful with the model, but in order to use the software in clinical practice, it requires a higher complexity to ensure a more accurate conversion.

Kurzzusammenfassung

Die Relevanz in der klinischen Praxis für den Transfer verschiedener MRT-Sequenzen liegt mitunter darin, dass die Zeit, die der Patient im Scanner verbringt, reduziert werden kann. Das Ziel der vorliegenden Bachelorarbeit ist die Evaluierung des diffusionsmodellbasierten Domänentransfers in der MRT-Bildgebung. Die Fragestellung lautet, ob das Modell die Sequenzkonvertierung in Bezug auf die Signaleigenschaften und in Bezug auf die Erhaltung der anatomischen Strukturen zwischen den realen und den generierten Bildern durchführen kann. Eine Diffusions-Autoencoder-Architektur in Kombination mit einem implementierten Kantenoptimierungsalgorithmus wird verwendet, sie wird an die Anforderungen medizinischer Bildanalyseverfahren angepasst und das Modell wird mit MRT-Sequenzen der unteren Extremitäten aus verschiedenen Datensätzen wie MRNet und FastMRI trainiert. Die Konvertierung funktioniert laut radiologischer Auswertung besser in Bezug auf die Signalkorrespondenz der konvertierten Bilder als auf die anatomische Korrespondenz der konvertierten Bilder. Außerdem funktioniert die Konvertierung nicht für jeden Sequenzübergang akzeptabel und die metrische Bewertung weicht von der radiologischen Bewertung ab. Zusammenfassend lässt sich sagen, dass die Konvertierung von MRT-Sequenzen mit dem Modell teilweise erfolgreich ist, aber um die Software in der klinischen Praxis einsetzen zu können, ist eine höhere Komplexität erforderlich, um eine genauere Konvertierung zu gewährleisten.

Contents

List of Abbreviations

MRI	Magnetic Resonance Imaging
T2w TSE	T2 weighted Turbuspine Echo Sequence
PDfs TSE	Proton Density fat saturated Turbospine Echo Sequence
PD TSE	Proton Density Turbospine Echo Sequence
COR	Coronal
CNN	Convolutional Neural Network
ROC	Receiver Operating Characteristic
AUC	Area Under Curve
ReLU	Rectified Linear Unit
ResBlock	Residual Block
Adam	Adaptive Moment Estimation
DDPM	Denoising Diffusion Probablistic Model
DDIM	Denoising Diffusion Implicit Model
DiffAE	Diffusion Autoencoder
GAN	Generative Adversarial Network
MSE	Mean Squared Error
MAE	Mean Absolute Error
FID	Fréchet Inception Distance
PSNR	Peak Signal to Noise Ratio
SSIM	Structural Similarity

List of Figures

List of Tables

Nomenclature

α	scalar value
$\boldsymbol{x}$	vector
$\boldsymbol{I}$	identity matrix
$\mathcal{N}(\boldsymbol{\mu}, \boldsymbol{\Sigma})$	multidimensional Gaussian distribution
$\mathcal{N}(\boldsymbol{x}_t; \boldsymbol{\mu}, \boldsymbol{\Sigma})$	multidimensional Gaussian distribution with output $\boldsymbol{x}_t$
$\mathbb{E}_{t,\boldsymbol{x}_0,\epsilon}$	expected value under the assumption t, $\boldsymbol{x}_0, \epsilon$
$\boldsymbol{\mu}_\theta$	neural network predicted mean
$\boldsymbol{\Sigma}_\theta$	neural network predicted variance
$\epsilon_\theta(\boldsymbol{x}_t, t, \boldsymbol{z}_{\text{sem}})$	neural network predicted noise conditioned by $\boldsymbol{x}_t$, t and $\boldsymbol{z}_{\text{sem}}$
$q(\boldsymbol{x}_t \vert \boldsymbol{x}_{t-1})$	forward diffusion process with output $\boldsymbol{x}_t$ and input $\boldsymbol{x}_{t-1}$
$q(\boldsymbol{x}_{t-1} \vert \boldsymbol{x}_t, \boldsymbol{x}_0)$	inference distribution conditioned by $\boldsymbol{x}_0$
$p_\theta(\boldsymbol{x}_{t-1} \vert \boldsymbol{x}_t)$	reverse diffusion process with output $\boldsymbol{x}_{t-1}$ and input $\boldsymbol{x}_t$
$p_\theta(\boldsymbol{x}_{t-1} \vert \boldsymbol{x}_t, \boldsymbol{z}_{\text{sem}})$	reverse diffusion process conditioned by $\boldsymbol{z}_{\text{sem}}$
L_{simple}	simplified loss expression
$tr(\boldsymbol{A})$	trace of matrix $\boldsymbol{A}$
$\sum_X$	covariance matrix of multidimensional Gaussian distribution X

Note

An important note beforehand, in the following we will prefer to use the term 'MRI sequence conversion' instead of 'domain transfer' from the title of the thesis because it describes the process more precisely.

1 Introduction

Torsional deformities of the lower limbs can affect joint health. Magnetic Resonance Imaging (MRI) images are used for diagnosis and manual measurements are taken by the radiologist. Since this manual measurement has a high standard deviation in practice due to a large number of different measurement approaches, the working group M&M – Machine Learning and Musculoskeletal Imaging of the Department of Diagnostic and Interventional Radiology of the University Hospital Aachen already developed a deep learning-based method to automate the torsion measurements [1]. Due to physics, if a radiologist wants to acquire multiple image data or sequences in the MRI scanner, he or she must scan the patient for a not inconsiderable amount of time for each new sequence. This is valuable time in clinical practice and the time in the scanner should be kept as short as possible for the patient.

The already implemented deep learning model is specifically conditioned to an axial T2 weighted Turbospine Echo Sequence [1]. However, since different MRI sequences are used in clinical practice, the model is not yet multifunctional. Therefore, another model which can perform MRI Sequence conversions is implemented.

The question, this thesis addresses, is whether the model will learn the MRI signal characteristics of different MRI sequences and subsequently whether it can perform MRI sequence conversions using a modified form of the Preechakul et al. [2] Diffusion Autoencoder. If the sequence conversion is successful, the output from the Diffusion Autoencoder then represents the input to the automated torsion measurement model.

The work is composed of two parts: In the first part, the focus is on the implementation of the Diffusion Autoencoder, as well as on the preprocessing of the training data and subsequently on the training of the model. Throughout the work, the focus is on refining the sequence conversion by implementing an Edge Optimization algorithm. In the second part, a clinical trial is conducted at the Department of Diagnostic and Interventional Radiology at the University Hospital Aachen. An internal test dataset of MRI image data is recorded with the size of 20 probands – ideally used for the evaluation of the model. However, during the work, it was realized that until the completion of the thesis, the use of the internal test data was not possible due to insufficient available training data.

The elaboration of the work is structured on the following pages as follows: First, an introduction to the medical background of MRI imaging is given and the measurement protocol used to acquire the test dataset is described. Second, the datasets used in this work are reviewed. Followed by the theory of Deep Learning, an explanation of the Diffusion Autoencoder and a presentation of the Edge Optimization algorithm. This is followed by the evaluation and discussion of the results which concludes in an outlook for follow-up work.

2 Medical Background

MRI sequences

MRI imaging is an imaging technique that does not require radioactive radiation. The method is based on the different amounts of hydrogen atoms in different regions of the body, which can be caused to spin by an externally applied magnetic field. First, a strong magnetic field is applied (typically in the range of 1.5 to 3 Tesla) which aligns the protons. The radiofrequency coils then stimulate the protons and deflect them out of their longitudinal and transverse equilibrium states. The signal is subsequently detected by the detector coils, which measure the energy released when the protons return to their equilibrium state. Different time intervals between excitation pulses and echo pulses and different atomic structures of the body's molecules lead to different contrasts of an MRI image. An MRI sequence is characterized by various parameters, primarily to control contrast, resolution, and highlighting of tissue structures via these parameters. MRI sequences that are relevant for this thesis will be discussed in the following paragraphs [3].

Turbo Spin Echo Sequence

In order to reduce the measurement time, the medical physicist Jürgen Henning from Freiburg devised a method in 1986 that made it possible to generate several spin echoes for one high frequency excitation pulse [3]. Conventionally, the single spin echo sequence can only generate one spin echo per repetition time T_{R} interval, which is shown in Figure 2.1. This corresponds to one line in the so-called k-space which is the Fourier transform of the measured image. The single line of k-space data is used to reconstruct a single image that corresponds to the time point at which the echo was acquired [4]. In a turbo spin echo pulse sequence, which is shown in Figure 2.1, multiple echoes are generated from one 90° radiofrequency excitation pulse, and each echo corresponds to a different line of k-space data. The k-space is therefore filled faster and more densely. This results in a faster and more detailed image acquisition. The 180° pulses rephase the spins so that it can be generated several echoes [3].

Spin-Lattice Relaxation (T1)

The spin lattice relaxation, also called T1 relaxation, describes the time period in which the aggregated proton spin becomes longitudinally aligned again. This process is exponential as shown in Figure 2.2(a). A T1 weighted MRI image results since different body tissues relax to the equilibrium state at different rates. For

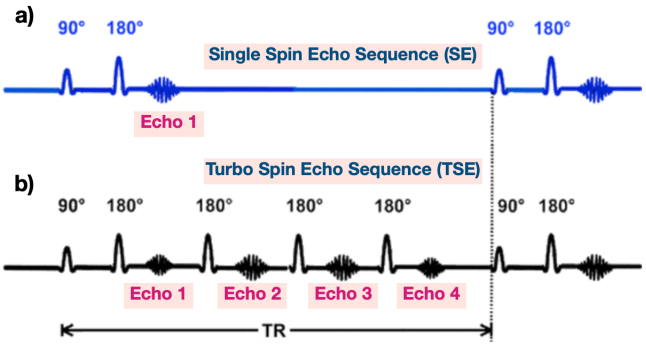
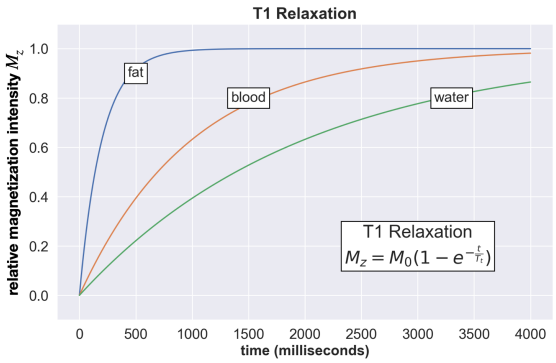
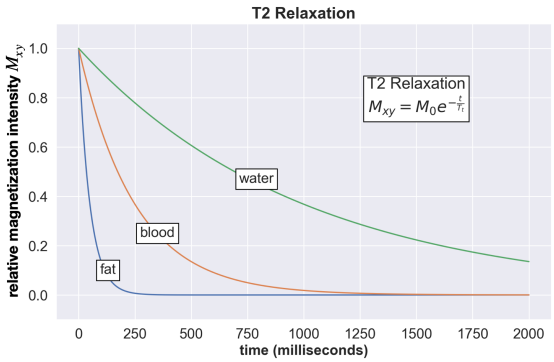


Figure 2.1: **a)** A Single Spin Echo sequence generated by a 90° RF pulse and a 180° rephasing pulse during the repetition time T_{R}.
b) A Turbo Spin Echo sequence with multiple echoes generated by recurring 180° rephasing pulses during the repetition time T_{R} [5].

example, fat is shown bright and fluids are shown dark because fat tissue realigns longitudinally significantly faster than fluids [3, 4].



(a) T1 relaxation times for fat, blood and water. The relative magnetization intensity M_z from 0.0 to 1.0 differs over time in milliseconds for the different tissues and converges exponential towards 1.0. It shows how fast the spin is longitudinally aligned again after excitation at t=0.



(b) T2 Relaxation times for fat, blood and water. The relative magnetization intensity M_{xy} from 0.0 to 1.0 differs over time in milliseconds for the different tissues and is converges exponential towards 0.0. It shows how fast the spin is transversely aligned again after excitation at t=0.

Figure 2.2: Illustration after [6]: (**a**) T1 relaxation time, (**b**) T2 relaxation time.

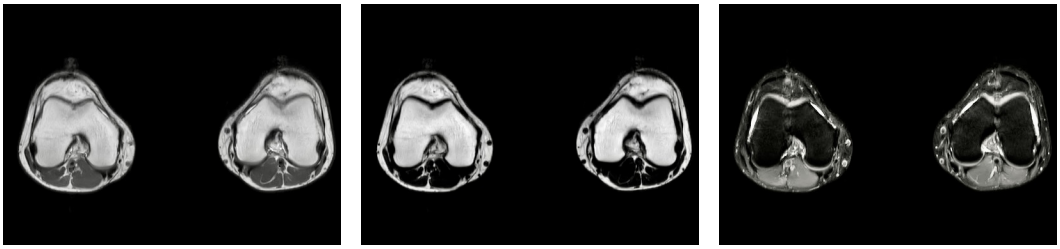
Spin-Spin Relaxation (T2)

The Spin-Spin relaxation, also known as T2 relaxation, describes the time period during which the decay of the transverse magnetization of the proton spin takes place. This process is also exponential as shown in Figure 2.2(b). The characteristic MRI image arises in this sequence due to the different properties of the body tissues. The more homogeneous a body tissue is, the longer the transverse magnetization decay takes because the magnetic moments of the individual spins

(spin-spin relaxation) affect each other less. For example, water is an extremely homogeneous tissue and therefore appears bright on the T2 MRI image [3].

Proton density weighted (PD) Dixon

The proton density weighted sequence maps the number of hydrogen protons in a volume element. Therefore, body tissues that have more protons per volume element appear brighter, e.g. fat or fluids [7].
The Dixon method can be combined with the proton density weighted sequence. The main goal is to obtain a fat suppression with the Dixon method. For this purpose, the so-called chemical shift is used, which is based on the different resonance frequencies of the tissue types. The different atomic structure of the fat and water molecules causes the spins of the protons to rotate at different speeds which results in different contrasts in the image. With one Dixon sequence a total of three subsequences is recorded. This consists of a fat suppression sequence in Figure 2.3(c), a water suppression sequence in Figure 2.3(b) and an inphase sequence (water + fat) in Figure 2.3(a) [7].



(a) Dixon inphase **(b)** Dixon water suppression **(c)** Dixon fat suppression

Figure 2.3: Different Dixon MRI sequences which differ in contrast.

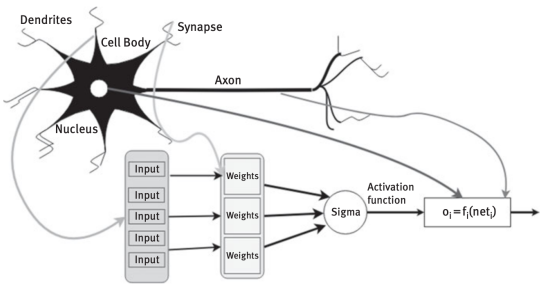
3 Machine Learning Theory

3.1 Deep Learning

Artificial Intelligence (AI) is the overall term for the models used in this thesis. An AI algorithm is designed to relieve humans of computationally intensive work and recognize patterns in large amounts of data. A subcategory of AI is machine learning (ML). The model for solving the problem is learned and adapted based on accumulated experience. Within ML there is the category of Deep Learning. However, ML is limited by the fact that humans must explicitly intervene in the learning process. Unsupervised Deep Learning models are trained with inputs without previously defined desirable outputs. Supervised Deep Learning models are trained with inputs connected to labels. To be successful with the unsupervised method, significantly larger datasets in relation to supervised datasets are needed. The layers and nodes inside a deep neural network are modeled like the human brain, or more precisely, the approximately 100 billion nerve cells. A comparison of the biological and artificial neuron is shown in Figure 3.1 [8, 9].

Biological neuron	Artificial neuron
Nerve cell	Neuron
Dendrites	Input signals
Synapses	Weights
Soma	Net input
Axon	Output

(a)



(b)

Figure 3.1: Comparison between a biological neuron and an artificial neuron. Input signals (dendrites) hit the input layer which corresponds to the soma. Whether the information contributes to the activation of the nerve cell and thus the transmission of information is decided by the synapses, which technically correspond to the weights and the activation function of the neural network. The connecting lines between the nerve cells are adjusted in the training process, which corresponds in the biological picture to the synapse knotting. The axon transfers the output to the next nerve cell [8].

Convolutional Neural Networks

Convolutional neural networks (CNN) are well adapted to the task of image recognition because they use the mathematical operation of convolution to compress the information contained in each pixel. Due to this operation, the model, unlike the regular neural network, has locally-connected layers, whose connections are also determined by the size of the kernel, and only at the end it has fully-connected layers [8].

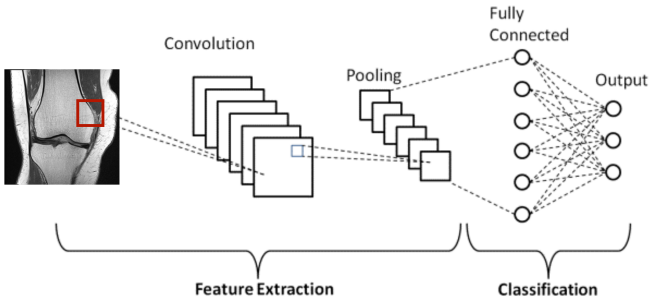


Figure 3.2: Illustration after [10]: A basic CNN with the kernel (red) performing the convolution on the image. Several convolutional layers with activation functions are followed by the pooling layers. The fully connected layers complete the architecture.

Usually a CNN block is composed of several layers that have different tasks, see Figure 3.2. The input layer receives the data of the individual pixels of an image. This is followed by one or more convolutional layers. These usually have several kernels, which are weighted differently, so that different information are recognized from the image. The recognition is performed by the activation layers which are frequently used as Rectified Linear Units (ReLU) or Sigmoid Functions, which both pass on information in principle, if a threshold value is exceeded. The detected features are then recompressed by one or more pooling layers. Common are max pooling and average pooling. These calculate the maximum or average value of a range of the feature matrix. Finally, the result of the calculation is computed by one or more fully-connected layers [8].

Residual Blocks (ResBlocks)

Residual blocks are applied in CNNs, especially if the network consists of several deep layers. The main task of the residual blocks is to monitor the gradients used to update the weights within the layer. These gradients should not become too small and not too large to ensure the learning ability of the model. Residual connections provide the solution, directly short-circuiting the input of the ResBlock to the output of the ResBlock or in multiple connections. This allows the neural network layers to calculate the difference between the true expected activation $\mathcal{T}(x)$ and the input x. So the layers in the ResBlock learn the residual $\mathcal{F}(x)$ and only at the end of a ResBlock the true output is calculated by using the Skip Connection, see

Figure 3.3 [11]. In Figure 3.7, the residual blocks used in the Diffusion Autoencoder are shown.

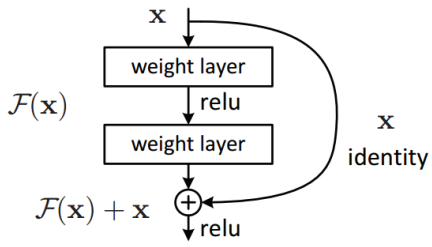


Figure 3.3: Residual Block with the input x, the layers which calculate the residual $\mathcal{F}(x)$ and use a rectified linear unit (ReLU) as activation and the true output $\mathcal{F}(x) + x$ [11].

$$\mathcal{F}(x) := Output - Input = \mathcal{T}(x) - x \tag{3.1}$$

$$\mathcal{T}(x) = \mathcal{F}(x) + x \tag{3.2}$$

Softmax Function

The softmax function or softmax layer is used to transform an input vector k of size N with $x \in \mathbb{R}$ values, into an output vector $\sigma(z)$ of size N with $x \in (0, 1)$ values. This transformation can be described as a probability transformation. Therefore, the softmax function is often used in multi-class classification problems. In Equation 3.3, the simple transformation prescription of the softmax function [12] is shown.

$$\sigma(k)_i := \frac{e^{k_i}}{\sum_{j=1}^{N} e^{k_j}} \quad i = 1, ..., N \tag{3.3}$$

Optimization and Loss Functions

The optimization of a neural network is performed by adjusting the weights and biases of the network so that the loss function is minimized. The optimization usually starts with the initialization of the weights and biases of the network. The network is then trained with the training data and in every iterative step the loss function is calculated. The error is then back-propagated and the weights and biases are adjusted using the optimization function to minimize the error. There are several optimization and loss functions that are used in practice – the L2 - Mean Squared Error, the L1 - Mean Absolute Error, the Cross-Entropy loss as well as the Adam Optimizer which are the used functions in the Diffusion Autoencoder, are discussed.

L2 - Mean Squared Error (MSE)

The Mean Squared Error [13] can be used both as a predictor and an estimator. As a predictor, the MSE measures the average squared difference between the predicted values $\hat{Y}_t$ and actual values Y_t, whereas $\hat{\boldsymbol{Y}}$ and $\boldsymbol{Y}$ can be a vector containing the pixel information of an image. The L2-loss is calculated over the whole vector of size n, see Equation 3.4.

$$\text{MSE} := \frac{1}{n}\sum_{t=1}^{n}\left(Y_t - \hat{Y}_t\right)^2 \tag{3.4}$$

As an estimator, the MSE estimates the expected value of the squared difference between the predicted distribution $\hat{\epsilon_\tau}$ and actual distribution ϵ_τ, based on a random chosen set of observed data with size n'.

$$\text{MSE}_{\hat{\epsilon_\tau}} := \frac{1}{n'}\sum_{\tau=1}^{n'}\mathbb{E}_{\epsilon_\tau}\left[\left(\epsilon_\tau - \hat{\epsilon_\tau}\right)^2\right] \tag{3.5}$$

The diffusion models are based on stochastic variables and calculations, so that Preechakul et al. [2] suggest the L2-loss MSE Estimator for the Diffusion Autoencoder training loss.

L1 - Mean Absolute Error (MAE)

The Mean Absolute Error [14] measures the average absolute difference between the prediction $\hat{Y}_t$ of the model and the actual values Y_t. The MAE loss is less susceptible to outliers because it does not increase quadratically. Outliers have only a linear influence on the loss as shown in Equation 3.6. The L1-loss is calculated over the whole vector of size n.

$$\text{MAE} := \frac{1}{n}\sum_{t=1}^{n}\left|Y_t - \hat{Y}_t\right| \tag{3.6}$$

As an estimator, the MAE estimates the expected value of the absolute difference between the predicted distribution $\hat{\epsilon_\tau}$ and actual distribution ϵ_τ, based on a random chosen set of observed data with size n'. Preechakul et al. [2] proposed the L1-loss MAE Estimator for the latent sampling training loss.

$$\text{MAE}_{\hat{\epsilon_\tau}} := \frac{1}{n'}\sum_{\tau=1}^{n'}\mathbb{E}_{\epsilon_\tau}\left[\left|\epsilon_\tau - \hat{\epsilon_\tau}\right|\right] \tag{3.7}$$

Cross-Entropy Loss

The Cross-Entropy loss function is often used in classification problems. The function receives the ground truth labels, which are either 0 for 'feature not included in image' and 1 for 'feature included in image', and the predictions of the classifier, which have been transformed by the softmax function to the range of values

between 0 and 1. The Cross-Entropy loss function measures the deviation between these two vectors and is given in Equation 3.8. A loss near 1 corresponds to a large difference and a loss near 0 corresponds to a small difference and thus a better performing classifier [15]. Preechakul et al. [2] proposed the Cross-Entropy loss for the Diffusion Autoencoder classifier training loss.

$$L_{\text{CE}} := -\sum_{i=1}^{n} \gamma_i log_{10}(p_i) \tag{3.8}$$

whereas n is the number of classes, γ_i are the truth labels and p_i are the predictions.

Adaptive Moment Estimation (Adam) Optimizer

Adam is an optimization algorithm to minimize the error in neural network training. Its goal is to find the optimal weights or parameters to make accurate predictions on new, independent data. Essentially, the Adam optimizer combines the advantages of two other optimization algorithms, the Stochastic Gradient Descent and the Root Mean Squared propagation optimizer. [16] The optimization prescription is as follows:

$$\Delta w_t := -\frac{\eta}{\sqrt{\hat{v}_t} + \epsilon} \hat{m}_t \tag{3.9}$$

whereas η is the learning rate, typically $\in [0.01, 0.0001]$, and ϵ a small constant to prevent dividing by zero.
However, Adam differs from other optimization algorithms by using adaptive moment estimators to adapt the size of the steps when updating the weights. This improves the convergence speed and stability of the training.

$$m_t := \beta_1 m_{t-1} + (1 - \beta_1) g_t \tag{3.10}$$

$$v_t := \beta_2 v_{t-1} + (1 - \beta_2) {g_t}^2 \tag{3.11}$$

$$\hat{m}_t := \frac{m_t}{1 - \beta_1^t} \tag{3.12}$$

$$\hat{v}_t := \frac{v_t}{1 - \beta_2^t} \tag{3.13}$$

whereas g_t is the gradient of the loss function and $\beta_1, \beta_2 \in [0, 1)$ to balance between the past weighting and adjust to current gradients.
The first moment estimator m_t in Equation 3.10 exists to compute the mean of the gradients, while the other moment estimator v_t in Equation 3.11 is used to estimate the variance of the gradients. In Equation 3.12 and Equation 3.13, the bias-corrected moment estimates which are included in Equation 3.9 to update the optimizer [16], are calculated.

UNet Architecture

The UNet architecture, first introduced by Ronneberger et al. [17] to segment medical images, consists of a contracting path and an expansive path which architecture is based on CNN blocks. The contracting path consists of several convolutional layers, each followed by ReLU layers and a MaxPooling layer that downsamples the input images. On each layer of the path, information of the image is extracted and transferred to the expanding path via Skip Connections. On the lowest level there is a compressed representation of the image, which is now upsampled by using Up-convolutional layers. Together with the information from the Skip Connections, it is the input for several Up-convolutional layers with subsequent ReLU layers. At the output of the UNet, a classification of each pixel is performed, which is understood as segmentation [17]. That UNet architecture in Figure 3.4 was implemented by the research group M&M - Machine Learning and Musculoskeletal Imaging to perform a segmentation of the lower limb torsion [1]. In this work, a modified UNet architecture is used which predicts the noise of an image. More information in the following Section 3.2 and in Figure 3.7

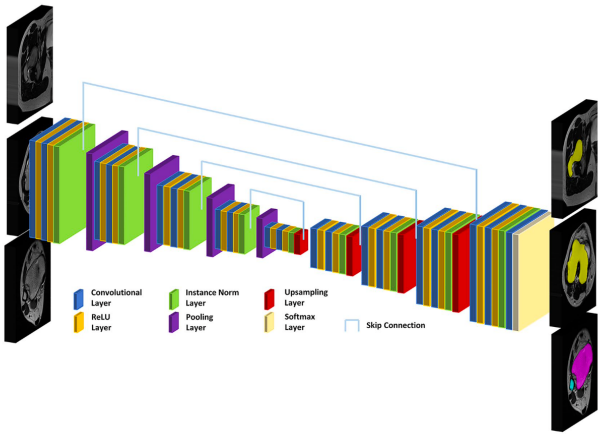


Figure 3.4: UNet CNN used for automatic segmentations by M&M [1]. It consists of a compressing part with Skip Connections which transfer local information from the Instance Norm Layer of each level to the Upsampling Layer of the decompressing part. Image information is extracted by using Convolutional Layers paired with ReLU activation. The Pooling Layers compress the information and Softmax function is used to produce a segmented output.

3.2 Diffusion Models

Diffusion models have become increasingly important in recent years as equivalent alternatives to Variational Autoencoders or Generative Adversarial Networks (GANs) in Generative AI. Deduced from Nonequilibrium Thermodynamics in physics by Sohl-Dickstein et al. [18], various models such as Denoising Diffusion Probablistic Models (DDPM) by Ho et al. [19] or Denoising Diffusion Implicit Models (DDIM) by Song et al. [20] have been developed. The basic idea of diffusion models is that a Markov Chain is performed. This mathematical model is used to describe the probability with which a system is in a certain state at a certain timestep and how it moves to another state with a certain probability at a later time.

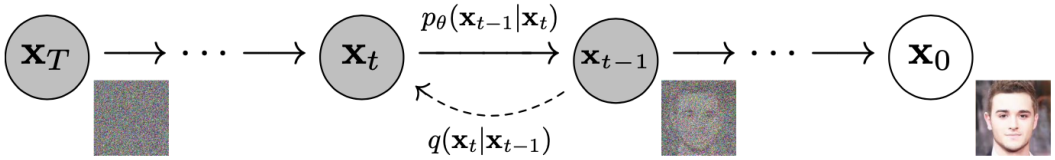


Figure 3.5: Forward diffusion process $q(\boldsymbol{x}_t|\boldsymbol{x}_{t-1})$ performed by adding gradually noise from original image $\boldsymbol{x}_0$ in steps t to the noisy image $\boldsymbol{x}_T$. Reverse diffusion process $p_\theta(\boldsymbol{x}_{t-1}|\boldsymbol{x}_t)$ performed by denoising images in every step t by predicting the noise $\epsilon_\theta(\boldsymbol{x}_t, t)$ until the original image $\boldsymbol{x}_0$ [19] is decoded.

Performing the Forward Diffusion Process shown in Figure 3.5, it is gradually added noise to the input image $\boldsymbol{x}_0$ in small steps t. T is typically chosen as $T = 1000$ which means that it is added a thousand times a small amount of noise to the image [19]. That affects how well the generated sample is approximated. The size of the steps is determined by the predetermined β scheduler $\beta_t \in (0, 1]$ which can be implemented in different ways. In the model, a linear scheduler is used which increases within $T = 1000$ steps of equal distance. The β scheduler shifts the mean $\sqrt{1-\beta_t}\boldsymbol{x}_{t-1}$ and the variance $\beta_t \boldsymbol{I}$ of the normal distribution in every step t. This generates noise. The forward process is defined as:

$$q(\boldsymbol{x}_1, ..., \boldsymbol{x}_T|\boldsymbol{x}_0) := \prod_{t=1}^{T} q(\boldsymbol{x}_t|\boldsymbol{x}_{t-1}) \tag{3.14}$$

$$q(\boldsymbol{x}_t|\boldsymbol{x}_{t-1}) := \mathcal{N}\Big(\boldsymbol{x}_t; \sqrt{1-\beta_t}\boldsymbol{x}_{t-1}, \beta_t \boldsymbol{I}\Big) \tag{3.15}$$

By using Bayes theorem, an expression for an inference distribution at time t is calculated that is conditioned by the input image $\boldsymbol{x}_0$. The inference distribution is in principle an estimation of the previous less noisy image $\boldsymbol{x}_{t-1}$ starting from $\boldsymbol{x}_t$. This expression is defined as the following:

$$q(\boldsymbol{x}_{t-1}|\boldsymbol{x}_t, \boldsymbol{x}_0) := \mathcal{N}\Big(\boldsymbol{x}_{t-1}; \boldsymbol{\mu}(\boldsymbol{x}_t, \boldsymbol{x}_0), \tilde{\beta}_t \boldsymbol{I}\Big) \tag{3.16}$$

$$\boldsymbol{\mu}(\boldsymbol{x}_t, \boldsymbol{x}_0) := \frac{\sqrt{\bar{\alpha}_{t-1}}\beta_t}{1-\bar{\alpha}_t}\boldsymbol{x}_0 + \frac{\sqrt{\alpha_t}(1-\bar{\alpha}_{t-1})}{1-\bar{\alpha}_t}\boldsymbol{x}_t \tag{3.17}$$

whereas $\alpha_t = 1 - \beta_t$, $\bar{\alpha}_t = \prod_{i=1}^{t} \alpha_i$ and $\tilde{\beta}_t := \frac{1-\bar{\alpha}_{t-1}}{1-\bar{\alpha}_t}\beta_t$.

If the exact reverse distribution $q(\boldsymbol{x}_{t-1}|\boldsymbol{x}_t)$ is known, a sample of the original image from $\boldsymbol{x}_T \sim \mathcal{N}(\mathbf{0}, \boldsymbol{I})$ (isotropic Gaussian distribution) can be generated. Since $\boldsymbol{x}_T \sim \mathcal{N}(\mathbf{0}, \boldsymbol{I})$ depends on the entire data distribution, a so-called reverse diffusion process is defined, also shown in Figure 3.5, which is trained with a deep neural network.

$$p_\theta(\boldsymbol{x}_0, ..., \boldsymbol{x}_T) := p(\boldsymbol{x}_T) \prod_{t=1}^{T} p_\theta(\boldsymbol{x}_{t-1}|\boldsymbol{x}_t) \tag{3.18}$$

$$p_\theta(\boldsymbol{x}_{t-1}|\boldsymbol{x}_t) := \mathcal{N}\Big(\boldsymbol{x}_{t-1}; \tilde{\boldsymbol{\mu}}_\theta(\boldsymbol{x}_t, t), \boldsymbol{\Sigma}_\theta(\boldsymbol{x}_t, t)\Big) \tag{3.19}$$

Specifically, the mean $\tilde{\boldsymbol{\mu}}_\theta(\boldsymbol{x}_t, t)$ is trained. With some mathematical reformulations the following expression is obtained.

$$\tilde{\boldsymbol{\mu}}_\theta(\boldsymbol{x}_t, t) = \frac{1}{\sqrt{\alpha_t}}\Big(\boldsymbol{x}_t - \frac{\beta_t}{\sqrt{1-\bar{\alpha}_t}}\epsilon_\theta(\boldsymbol{x}_t, t)\Big) \tag{3.20}$$

whereas $\epsilon_\theta(\boldsymbol{x}_t, t)$ is an approximation to predict the noise ϵ from the sample $\boldsymbol{x}_t$. The training is done by learning the predicted noise $\epsilon_\theta(\boldsymbol{x}_t, t)$.

With these two expressions, the exact mean $\boldsymbol{\mu}(\boldsymbol{x}_t, \boldsymbol{x}_0)$ and the predicted mean $\tilde{\boldsymbol{\mu}}_\theta(\boldsymbol{x}_t, t)$, the loss function from Equation 3.4, that indicates how well the diffusion model performs, is defined. This function calculates the deviation between the predicted noise of the sample and the actual noise of the sample. The calculation of the loss function is abbreviated using the results of Ho et al. [19] and the simplified loss objective (L2-loss) of this function is given.

$$L_{\text{simple}} := \mathbb{E}_{t,\boldsymbol{x}_0,\boldsymbol{\epsilon}}\Big[\|\epsilon - \epsilon_\theta(\sqrt{\bar{\alpha}_t}\boldsymbol{x}_0 + \sqrt{1-\bar{\alpha}_t}\epsilon, t)\|^2\Big] \tag{3.21}$$

Denoising Diffusion Implicit Model (DDIM)

Denoising Diffusion Implicit Models (DDIMs) by Song et al. [20] differ from Denoising Diffusion Probablistic Models (DDPMs) by providing a more efficient way to generate samples. The basis of the training process remains the same, but instead of the reverse Markov chain in DDPMs, non-Markovian deterministic processes are used in the DDIM architecture. As a result, DDPM and DDIM are both similar in the loss function as well as in the marginal distribution and only differ in the way the samples are generated. Equation 3.23 and Equation 3.24 show how the samples are generated for DDPMs and DDIMs. These expressions result from the inference distribution of both models which slightly differ as shown in Equation 3.16 for DDPMs and in Equation 3.22 for DDIMs [2, 19].

$$q_{(\boldsymbol{x}_{t-1}|\boldsymbol{x}_t,\boldsymbol{x}_0)_{\text{DDIM}}} := \mathcal{N}\Big(\sqrt{\alpha_{t-1}}\boldsymbol{x}_0 + \sqrt{1-\alpha_{t-1}}\frac{\boldsymbol{x}_t - \sqrt{\alpha_t}\boldsymbol{x}_0}{\sqrt{1-\alpha_t}}, \mathbf{0}\Big) \tag{3.22}$$

$$\boldsymbol{x}_{t-1_{\text{DDPM}}} := \frac{1}{\sqrt{\alpha_t}}\Big(\boldsymbol{x}_t - \frac{1-\alpha_t}{\sqrt{1-\bar{\alpha}_t}}\epsilon_\theta(\boldsymbol{x}_t, t)\Big) + \sigma_t\boldsymbol{\Omega} \tag{3.23}$$

$$\boldsymbol{x}_{t-1_{\text{DDIM}}} := \sqrt{\alpha_{t-1}}\underbrace{\Big(\frac{\boldsymbol{x}_t - \sqrt{1-\alpha_t}\epsilon_\theta(\boldsymbol{x}_t)}{\sqrt{\alpha_t}}\Big)}_{:=f_\theta(x_t)} + \sqrt{1-\alpha_{t-1}-\sigma_t^2}\epsilon_\theta(\boldsymbol{x}_t) + \sigma_t\boldsymbol{\Omega} \tag{3.24}$$

whereas $\boldsymbol{\Omega} \sim \mathcal{N}(\mathbf{0}, \boldsymbol{I})$ and $f_\theta(\boldsymbol{x}_t)$ is the predicted $\boldsymbol{x}_0$.
Assuming $\sigma_t = 0$, the model becomes an implicit probablistic model with the samples of $\boldsymbol{x}_T$ to $\boldsymbol{x}_0$ generated by the objective of the DDPM, but the process is deterministic and not diffusive. This basically means that the model is consistent so that multiple generated samples from the DDIM latent code have similar high level features [2, 19, 20].

3.2.1 Diffusion Autoencoder

The Diffusion Autoencoder (DiffAE) by Preechakul et al. [2] combines several concepts and models of deep learning networks and diffusion models. This architecture convinces in the almost perfect reconstruction of input images, in the semantic interpolation between two input images, in the generation of completely new images, as well as in the attribute manipulation of images. In Figure 3.6 the superficial structure of DiffAE is given, which is described in detail in the following. The noise predicting network inside the Conditional DDIM, shown in Figure 3.7, is implemented based on a modified UNet architecture from Dhariwal et al. [21].

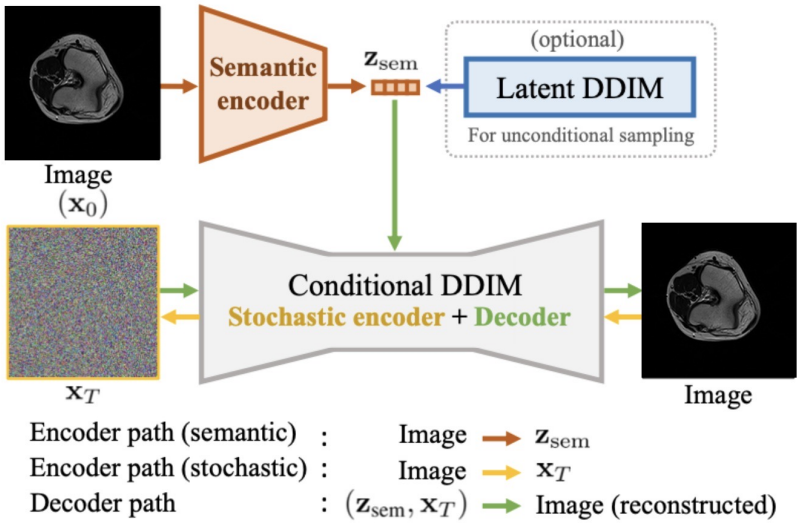


Figure 3.6: Overview of the Diffusion Autoencoder by Preechakul et al. [2]. The autoencoder consists of a semantic encoder that encodes the input image $\boldsymbol{x}_0$ to the semantic subcode $\boldsymbol{z}_{\text{sem}}$ and a conditional DDIM that both encodes $\boldsymbol{x}_0$ to the stochastic subcode $\boldsymbol{x}_T$ and decodes $\boldsymbol{x}_T$ conditioned by $\boldsymbol{z}_{\text{sem}}$ to the output $\boldsymbol{x}_0$. A latent DDIM is trained to sample from $\boldsymbol{x}_T \sim \mathcal{N}(0, \boldsymbol{I})$ conditioned by $\boldsymbol{z}_{\text{sem}}$. The actual training to predict the noise $\epsilon_\theta(\boldsymbol{x}_t, t, \boldsymbol{z}_{\text{sem}})$ with a UNet is performed in the decoder path.

Semantic Encoder

The semantic encoder is used to compress the input image $\boldsymbol{x}_0$ into a latent vector $\boldsymbol{z}_{\text{sem}}$ to learn the semantic features. To compress the input image into this vector $\boldsymbol{z}_{\text{sem}}$ with dimensions 1x512, the encoder path of the denoising UNet is used, which is shown in Figure 3.7 on the right. The latent information contained in $\boldsymbol{z}_{\text{sem}}$ is given as input to the decoder from Section 3.2.1 of the conditional DDIM, thus the DDIM is conditioned with valuable latent information about $\boldsymbol{x}_0$ and accelerate the denoising diffusion process. Moreover, the decoder $p_\theta(\boldsymbol{x}_{t-1}|\boldsymbol{x}_t, \boldsymbol{z}_{\text{sem}})$ of the DDIM receives the stochastic subcode $\boldsymbol{x}_T$.

Stochastic Encoder

The stochastic encoder is based on the DDIM principles described earlier. A given input image $\boldsymbol{x}_0$ is stochastically encoded and the process steps of non-Markovian deterministic processes are performed until the stochastic subcode $\boldsymbol{x}_T$ is obtained. A lot of information about the input image is already stored in $\boldsymbol{z}_{\text{sem}}$ and just details in $\boldsymbol{x}_T$ are encoded that are not considered in $\boldsymbol{z}_{\text{sem}}$. To do this, Equation 3.24 is used and the prediction function $f_\theta(\boldsymbol{x}_t)$ is reparametrized by modifying $\epsilon_\theta(\boldsymbol{x}_t)$ to $\epsilon_\theta(\boldsymbol{x}_t, t, \boldsymbol{z}_{\text{sem}})$.

$$f_\theta(\boldsymbol{x}_t, t, \boldsymbol{z}_{\text{sem}}) := \Big(\frac{\boldsymbol{x}_t - \sqrt{1-\alpha_t}\epsilon_\theta(\boldsymbol{x}_t, t, \boldsymbol{z}_{\text{sem}})}{\sqrt{\alpha_t}}\Big) \tag{3.25}$$

With this expression, it can be understood how to determine the $\boldsymbol{x}_T$ with $\boldsymbol{z}_{\text{sem}}$ taken into consideration. To do this, the generative process from Equation 3.24 is reversed, as shown in Equation 3.26.

$$\boldsymbol{x}_{t+1_{\text{DiffAE}}} := \sqrt{\alpha_{t+1}} f_\theta(\boldsymbol{x}_t, t, \boldsymbol{z}_{\text{sem}}) + \sqrt{1-\alpha_{t+1}}\epsilon_\theta(\boldsymbol{x}_t, t, \boldsymbol{z}_{\text{sem}}) \tag{3.26}$$

Decoder

As mentioned, the decoder of the conditional DDIM receives as input both the semantic subcode $\boldsymbol{z}_{\text{sem}}$ and the stochastic subcode $\boldsymbol{x}_T$. To understand the generative process of the decoder, the reverse diffusion process from Equation 3.19 which is extended by the conditioning with $\boldsymbol{z}_{\text{sem}}$ is important:

$$p_\theta(\boldsymbol{x}_0, ..., \boldsymbol{x}_T | \boldsymbol{z}_{\text{sem}}) := p(\boldsymbol{x}_T) \prod_{t=1}^{T} p_\theta(\boldsymbol{x}_{t-1} | \boldsymbol{x}_t, \boldsymbol{z}_{\text{sem}}) \tag{3.27}$$

In Section 3.2 the idea is presented that first the inference distribution $q(\boldsymbol{x}_{t-1}|\boldsymbol{x}_t, \boldsymbol{x}_0)$ and second the predicted distribution contribute to the loss function. $q(\boldsymbol{x}_{t-1}|\boldsymbol{x}_t, \boldsymbol{x}_0)$ is the ground truth and p_θ approximates the distribution. The composition of $p_\theta(\boldsymbol{x}_{t-1}|\boldsymbol{x}_t, \boldsymbol{z}_{\text{sem}})$ is defined in Equation 3.28.

$$p_\theta(\boldsymbol{x}_{t-1}|\boldsymbol{x}_t, \boldsymbol{z}_{\text{sem}}) := \begin{cases} \mathcal{N}(f_\theta(\boldsymbol{x}_1, 1, \boldsymbol{z}_{\text{sem}}), 0) & \text{if } t = 1 \\ q(\boldsymbol{x}_{t-1}|\boldsymbol{x}_t, f_\theta(\boldsymbol{x}_t, t, \boldsymbol{z}_{\text{sem}})) & \text{otherwise} \end{cases} \tag{3.28}$$

The function f_θ, see Equation 3.25, is the approximation of the ground truth image $\boldsymbol{x}_0$. With this in mind, the interaction between the forward process q and the generative process p_θ can be better understood. Finally, the loss calculation between the actual noise ϵ_t and the predicted noise $\epsilon_\theta(\boldsymbol{x}_t, t, \boldsymbol{z}_{\text{sem}})$ over a large number n is calculated, e.g. 1000. By minimizing the loss, the denoising UNet learns to predict the noise of an image x_t.

$$L_{\text{simple}} := \sum_{t=1}^{n} \mathbb{E}_{\boldsymbol{x}_0, \epsilon_t}\Big[\|\epsilon_t - \epsilon_\theta(\sqrt{\alpha}_t \boldsymbol{x}_0 + \sqrt{1-\alpha_t}\epsilon_t, t, \boldsymbol{z}_{\text{sem}})\|_2^2\Big] \tag{3.29}$$

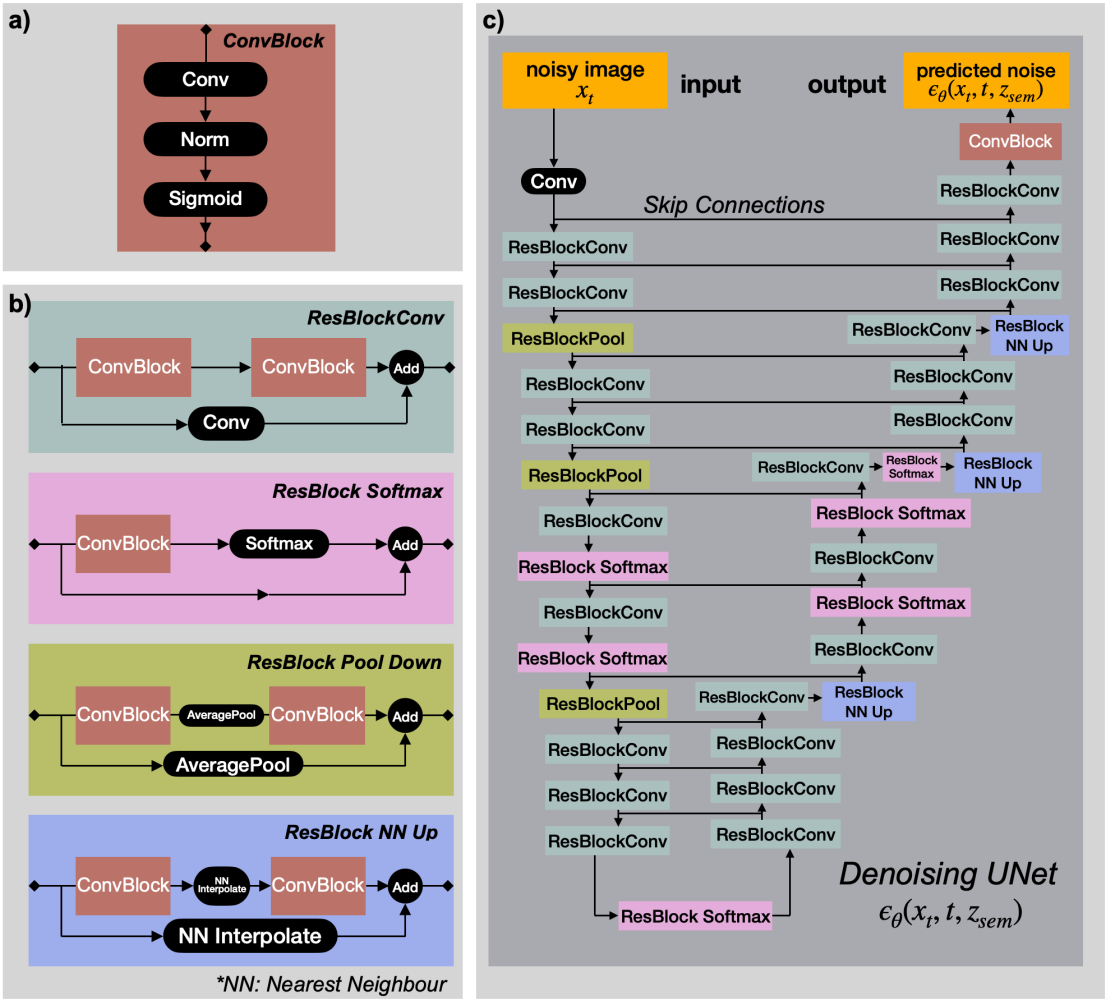


Figure 3.7: UNet architecture of the DiffAE model.
a) One ConvBlock consists of a convolutional Layer (3x3 kernel size), an Instance Normalization Layer and a Sigmoid activation function.
b) A standard ResBlock consists of two ConvBlocks and a convolutional Skip connection. ResBlock Softmax to normalize midlayer output in a probability distribution. Average Pooling is used for downsampling and Nearest Neighbour Interpolation is used for upsampling.
c) The denoising UNet architecture consists of the encoder path on the left which compresses a noisy image $\boldsymbol{x}_t$ and transfer local information on each level to the decoder path on the right via Skip Connections. The output is the noise prediction $\epsilon_\theta(\boldsymbol{x}_t, t, \boldsymbol{z}_{\text{sem}})$ of the image $\boldsymbol{x}_t$ at step t conditioned by the high-level semantics $\boldsymbol{z}_{\text{sem}}$.

3.3 Classifier

Classifying images, text or objects is a large area of machine learning. There exist different approaches to optimize the classifier. A basic linear classifier that uses logistic regression is trained using a Cross-Entropy loss. The model of the linear classifier expects two parameters as initialization. The first one is the number of features of the input vector and the second one is the number of classes of which the dataset consists. In one training step, the DiffAE classifier receives the features, which are contained in the latent vector $\boldsymbol{z}_{\text{sem}}$, and applies a linear transformation to it, as can be seen in Equation 3.30.

$$f(x) := \boldsymbol{w}^T \boldsymbol{x} + b \tag{3.30}$$

whereas w_i are the weights to be adjusted in the learning process, x_i are the features of $\boldsymbol{z}_{\text{sem}}$ and b is a bias.

The calculation of $f(x)$ is done for each of the previously specified classes and then the logistic regression is performed using the softmax function. Thus, a vector of the size of the number of classes is obtained, which contains the probabilities for each class. With the help of the Cross-Entropy loss, the deviation from the ground truth is calculated and then the weights of the linear classifier are updated by the optimizer. Figure 3.8 shows a visualization of a multiclass classifier. The figure demonstrates how datapoints get classified in a two or higher dimensional space [22].

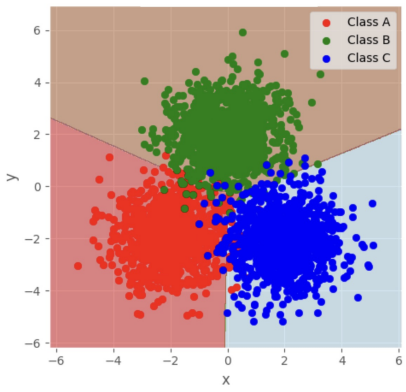


Figure 3.8: A multiclass classifier visualization of three classes A, B and C with partition of the two dimensional space in three areas. Datapoints within one area are classified to the related class because they consist of similar properties [23].

The performance of a classifier is measured by calculating the Receiver Operating Characteristic (ROC) curve. The ROC curve represents the rate of true positive outcomes versus the rate of false positives. These can be easily calculated according to Equation 3.31 and according to Equation 3.32. After both rates have been plotted on a graph as in Figure 3.9, a curve can be calculated from them.

A diagonal line between the point $(0, 0)$ and $(100, 100)$ corresponds to a random classification and a curve through the point $(0, 100)$ indicates a perfect classification. In simple classification tasks, such perfect performance may occur. The area under the curve is called the Area Under Curve (AUC) and gives a value for the percentage performance of the classifier [24].

$$\text{TruePositiveRate} := \frac{\text{TruePositives}}{\text{TruePositives} + \text{FalseNegatives}} \tag{3.31}$$

$$\text{FalsePositiveRate} := \frac{\text{FalsePositives}}{\text{FalsePositives} + \text{TrueNegatives}} \tag{3.32}$$

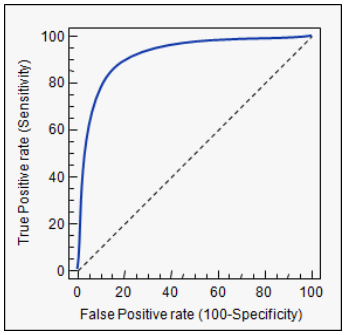


Figure 3.9: ROC Curve for a binary classifier. The steeper the curve, the higher is the true positive rate and the better the classifier. The dashed line is a random classification with exactly the same true positive rate and false positive rate [25].

Moreover, the performance of a classifier can be displayed in a confusion matrix, see Figure 3.10. This is basically a four- or multiple-field table and quickly gives a good overview of how much test data has been classified correctly and how much has been classified incorrectly. Both metrics can be applied to multiclass classifiers.

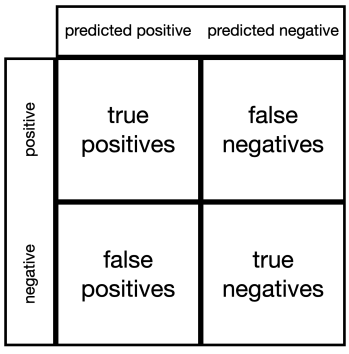


Figure 3.10: A confusion matrix for a binary classifier. The columns represent the prediction positive or negative and due to the actual label positive or negative in the rows, one classification can be counted to one of the four fields [24].

3.4 Edge Detection

Images have many properties like contrast, brightness or color. But in addition, images usually map an object or something similar. One of the big topics in the field of computer vision and also medical image analysis is segmentation. A simple form of segmentation is the detection of the edges of an object. John F. Canny developed the so-called Canny Edge detection algorithm in 1986 [26]. This contains several filter operations which are applied to an image, so that finally an image with black and white points is created, where white points stand for the edges of an image. Figure 3.11 shows an MRI image to which the Canny Edge detection algorithm has been applied.

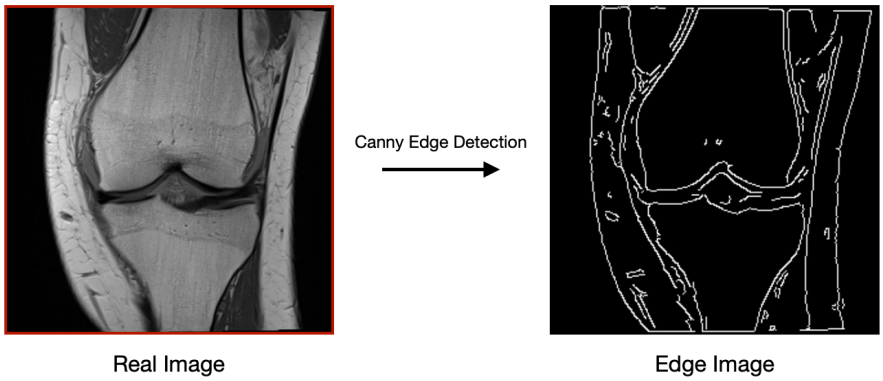


Figure 3.11: Canny Edge Detection algorithm from OpenCV [27] performed on a COR PD MRI sequence. The edge image shows the contours of bone, tissue and the entire object.

The following is a brief explanation of the algorithm's operations. In a first step, the original image is smoothed by applying a noise reduction filter. Then the gradient is calculated pixel by pixel using the Sobel operator in both horizontal and vertical direction. This corresponds approximately to a human perception of how strong the contrast between light and dark. In a third step, the local maxima of the gradients are searched for. First it is checked whether it is a local maximum and then it is searched in the direction of the gradient whether an adjacent gradient still has a larger maximum. The weaker gradients are then set to 0. At last an evaluation follows, whether it is an edge or not, by using a hysteresis thresholding procedure. One specifies a minimum and maximum value. If the gradient is below the minimum value, it is classified as no edge, if the gradient is above the maximum value, the gradient is considered an edge. The gradients within the interval are interesting. Only those gradients are classified as edges that are contiguous and directly connected to a gradient that is above the maximum threshold value [26,27].

4 Datasets

The performance of Deep Learning Models depends considerably on the quality and the amount of available training samples [28]. Datasets containing different MRI sequences of the lower limb are required. The MRNet [29] dataset is used as well as the data from the fastMRI [30] dataset. A summary of the training dataset and test dataset can be found in Table 5.2 and Table 5.5. Furthermore, the internal test dataset of axial images is recorded. Due to insufficient available training datasets of axial MRI images of the lower limb region, the internal test dataset can not be used in this thesis. In follow-up work, the acquired data can serve as useful test data to successfully integrate the sequence conversion model into the model for torsion segmentation. In this work, the focus is on the proof of concept and therefore the MRI sequences COR PD, COR PD FS and COR T1 are chosen.

4.1 MRNet by StandfordML

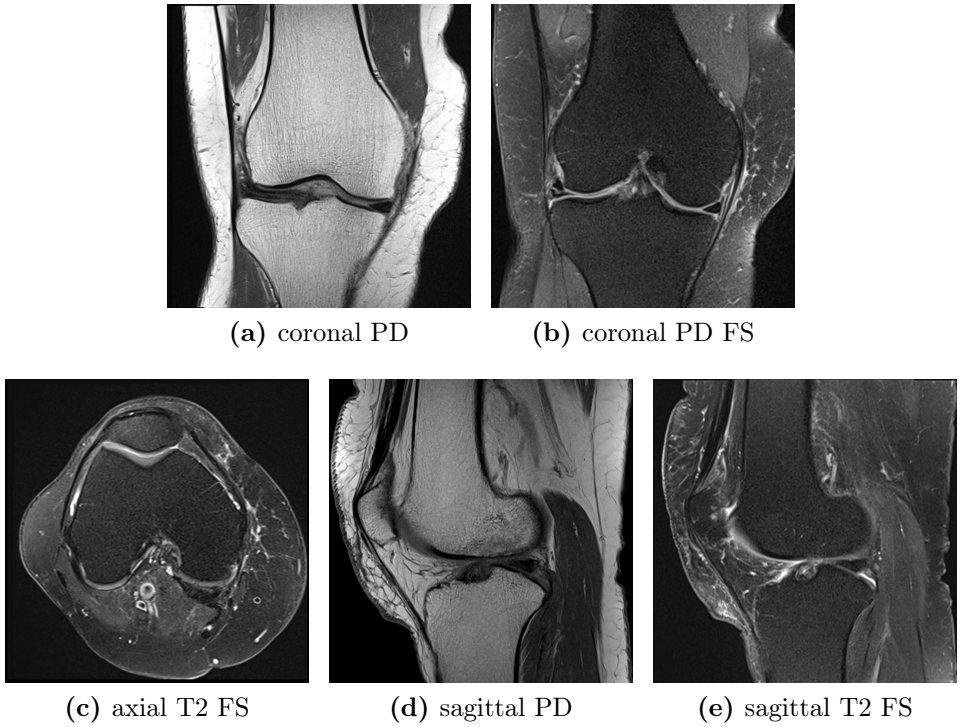
The MRNet dataset [29] contains a total of 1.370 MRI scans of the knee. One and the same knee was scanned axially, coronally and saggitally. Furthermore, the different planes were imaged in different sequences. In the coronal slice, the MRI scan was imaged with T1 weighting, see Figure 4.1(a), in the saggital slice T2 weighting with fat saturation, see Figure 4.1(b) and in the axial slice proton density weighted with fat saturation, see Figure 4.1(c) [29].
Ordinarily, researchers at the Standford ML Group splitted the dataset into a training dataset (1.130 images), test dataset (120 images), and a validation dataset (120 images). A split of the data is not needed for the task of this thesis. – the entire dataset is used as training data for the Diffusion Autoencoder.

(a) coronal T1 **(b)** sagittal T2 FS **(c)** axial PD FS

Figure 4.1: MRNet knee MRI sequences in three sectional planes and three different contrasts [29].

4.2 fastMRI by NYU

The fastMRI dataset [30] from the department of Radiology at NYU School of Medicine and NYU Langone Health consists of 10.000 clinical knee MRI images. In the coronal plane, the MRI scan was imaged with proton density-weighted with and without fat suppression, see Figure 4.2(a) and Figure 4.2(b), in the axial plane proton density-weighted with fat suppression, see Figure 4.2(c) and in the sagittal plane proton density-weighted and T2-weighted with fat suppression, see Figure 4.2(d) and Figure 4.2(e) [30].



(a) coronal PD **(b)** coronal PD FS

(c) axial T2 FS **(d)** sagittal PD **(e)** sagittal T2 FS

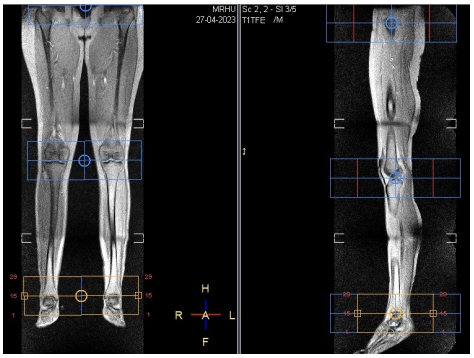
Figure 4.2: fastMRI knee MRI sequences in three sectional planes and three different contrasts [30].

4.3 Internal MRI dataset

The measurements are obtained on the MRI scanner at the Department of Diagnostic and Interventional Radiology of the University Hospital Aachen, as shown in Figure 4.3(a). A Philips Ingenia Elition 3.0T X MRI scanner is used there. The study includes a size of N = 20 probands, male and female between 20 and 30 years of age.

The measurement protocol consists of the following steps: First, the proper positioning of the proband is adjusted in the MRI scanner. The three body regions (hip joint, knee joint and ankle joint) are positioned centrally within the Field of View in the axial, coronal and sagittal planes, as shown in Figure 4.3(b). Subsequently,

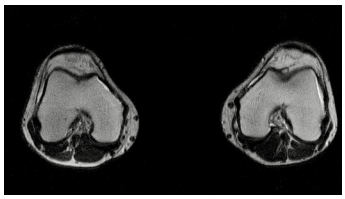
(a) MRI scanner in a protected room with the large body coil and the mobile couch, which positions the proband inside the coil.



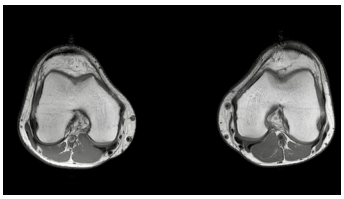
(b) Positioning the Field of View in the hip, knee and ankle region to make sure that the scanner scans the relevant regions.

Figure 4.3: MRI Measurements at University Hospital Aachen.

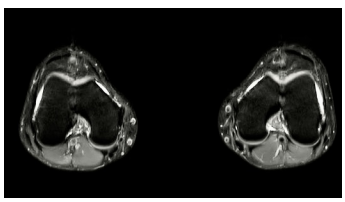
the first sequence is recorded. This will later serve as the reference frequency, as it is recorded with the modalities, such that it can serve as the input image for the automatic torsion measurement model of the M&M research team [1]. The name of the sequence is 4mm T2w axial 3Stacks Std, see Figure 4.4(a), which means a T2 sequence in the axial direction is used, 3 stacks corresponding to the three body regions (hip, knee and ankle) are acquired, and a slice thickness is chosen, which images the interior of the body, of 4 mm. The acquisition of this sequence has a total scan duration of 05:50 minutes. The reference sequence is now followed by the sequences 4mm T1w axial 3Stacks, shown in Figure 4.4(b) as well as 4mm PDfs axial 3Stacks mDixon. The fat suppression and water suppression sequences from the mDixon sequence are shown in Figure 4.4(c) and Figure 4.4(d).



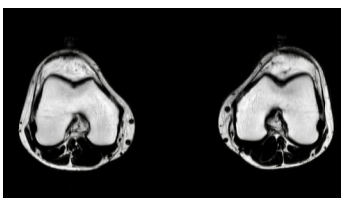
(a) axial T2w



(b) axial T1w



(c) axial PD fat supp.



(d) axial PD water supp.

Figure 4.4: Empirical MRI sequence data from the axial scanned knee. The different contrasts appear in the area of the bones and the tissue.

5 Methods

In the following Section, the implementation of the Diffusion Autoencoder performing the MRI sequence conversion is elaborated. Therefore some common metrics are introduced in Section 5.4. To get a good overview of how the training of the DiffAE model is structured, see Figure 5.1. The DiffAE architecture is presented first, the classifier parameters are provided second, then the Edge Optimization algorithm is introduced and finally the implementation of several metrics is presented.

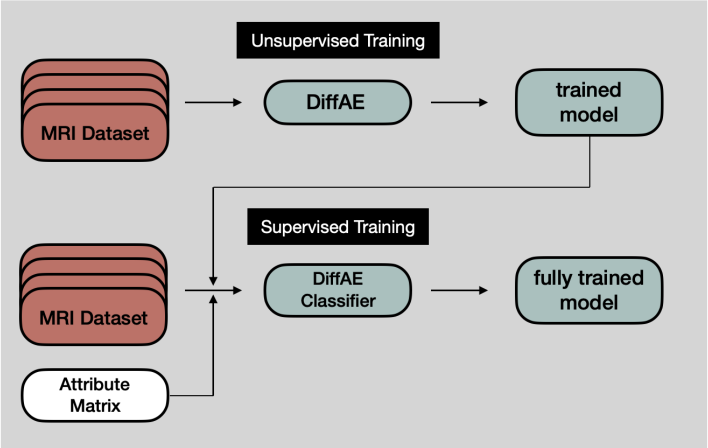


Figure 5.1: Overview of the training procedure. First the DiffAE is trained unsupervised with the training dataset. The trained model is able to encode and decode but it needs supervised information about the MRI sequences. Therefore, the same training data is encoded into their latent vectors $\boldsymbol{z}_{\text{sem}}$ by the trained model. Every $\boldsymbol{z}_{\text{sem}}$ is labeled by the attribute matrix so that the classifier can be trained. The fully trained model performs the MRI sequence conversion by shifting the latent vectors in latent space.

5.1 Diffusion Autoencoder

A tabular overview of the training parameters of the Diffusion Autoencoder is provided in Table 5.1 and a precise indication of the number of each training data is provided in Table 5.2.

Since the training data is provided by different institutions, they differ in resolution and especially in file formats. The MRI scans are converted to a uniform resolution of 128x128 pixels and the most informative slices from each of the MRI

Table 5.1: Network architecture and training parameters of the Diffusion Autoencoder for the MRI Knee COR training dataset with 35.635 three channel RGB MRI images. The final training took 132 hours on two NVIDIA Tesla Volta V100 GPUs with 20.000.000 entered samples and a batch size of 32. Other parameters are related to the used loss, optimizer and DiffAE architecture parameters

Parameter	**MRI Knee COR**
Batch size	32
Images trained	20.000.000
Total train dataset	35.635
Image color space	three channel RGB
Base channels	128
Channel multipliers	[1,1,2,3,4]
Attention resolution	[16]
Encoder base channels	128
Encoder channel multipliers	[1,1,2,3,4,4]
Encoder attention resolution	[16]
$\mathbf{z}_{\text{sem}}$ size	512
β scheduler	Linear
Learning rate	1e-4
Optimizer	Adam (no weight decay)
Training T	1000
Diffusion loss	MSE with noise prediction ϵ
Num. GPUs	2
Absolute training time	$\sim$ 132h

scans are extracted. Moreover, the one channel grayscale MRI images are converted to three channel RGB MRI images with the same pixel information in the three channels. This is due to the existing network architecture. Improvements through reducing the input channels of the Diffusion Autoencoder are discussed in Section 7.

Table 5.2: MRI Knee COR training dataset split

Dataset	**Organisation**	**Sequence**	**Number**
fastMRI	NYU	COR PD	14.815
fastMRI	NYU	COR PD FS	15.368
MRNet	StandfordML	COR T1	5.452

To generalize the Diffusion Autoencoder, multiple augmentations are applied to the MRI training dataset, which are specified in table 5.3. This is common practice in Deep Learning and results in the model being able to handle a wider variation of images. The transforms tool from the torchvision library is used. The whole training dataset consists of COR PD, COR PD FS as well as COR T1 sequences and the augmentations such as horizontal flips, vertical flips, Gaussian Noise, rotations by a few degrees around the center perpendicular of the image as well as affine

shifts of the images by maximum 20% of the image width and height are shown in Table 5.3 and in Figure 5.2.

Table 5.3: Image augmentations specified with their range or value and their probability of use from the torchvision library [31]

Augmentations	**Range/Value**	**Probability of use**
Random Rotation	(-10°, 10°)	100%
Random Horizontal Flip	-	50%
Random Vertical Flip	-	50%
Random Affine	$x = (-0.2, 0.2), y = (-0.2, 0.2)$	100%
Random Gaussian Noise	$mean = 0, variance = 0.05$	50%

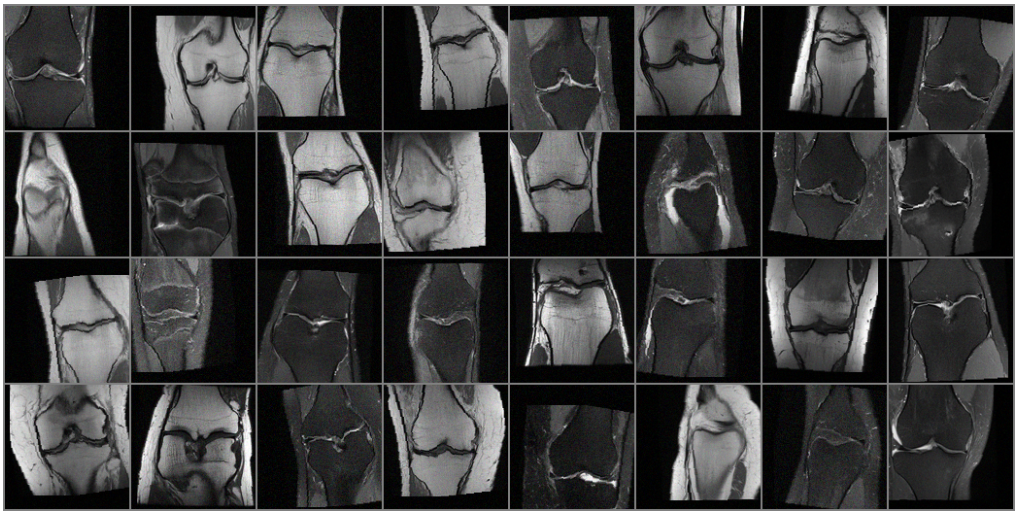


Figure 5.2: A batch of 32 real MRI images with COR PD, COR PD FS and COR T1 sequences as input for the Diffusion Autoencoder. The used augmentations are applied.

5.2 Classifier and Sequence Conversion

The sequence conversion itself is performed by shifting the latent vector $\boldsymbol{z}_{\text{sem}}$ along a linear straight by the linear Classifier. Therefore, the classifier is trained on the same training dataset used for the Diffusion Autoencoder training. But the classifier receives a tuple consisting of the encoded $\boldsymbol{z}_{\text{sem}}$ by the trained DiffAE and a label e.g. 'COR PD' by the attribute matrix. The linear classifier fills the latent space with a $\boldsymbol{z}_{\text{sem}}$ belonging to the sequence name. If the training is finished, the latent space is divided by the classifier in three areas corresponding to the three possible MRI sequences. A tabular overview of the training parameters for the training of the classifier is provided in Table 5.4.

Table 5.4: Training parameters of the linear classifier training for the MRI Knee COR Classifier training dataset with the same but now labeled 35.635 images. The final training took 1 hour on one NVIDIA Tesla Volta V100 GPU with 150.000 entered samples and a batch size of 2. Other parameters are related to the used loss and the learning rate

Parameter	**MRI Knee COR Classifier**
Batch size	2
Images trained	150.000
Total train dataset	35.635
Learning rate	1e-5
Diffusion loss	Cross Entropy Loss
Num. GPUs	1
Absolute training time	$\sim$ 1h

The classification performance is tested by calculating the Receiver Operating Characteristic (ROC) curve and the confusion matrix on a test dataset. A detailed description of the test data is listed in Table 5.5.

Table 5.5: MRI Knee COR Classifier test dataset split

Dataset	**Organisation**	**Sequence**	**Number**
fastMRI	NYU	COR PD	416
fastMRI	NYU	COR PD FS	448
MRNet	StandfordML	COR T1	478

The sequence conversion is performed by shifting the $\boldsymbol{z}_{\text{sem.original}}$, encoded from e.g. a 'COR PD' sequence, along a straight in the latent space. Therefore the classifier receives the sequence name e.g. 'COR T1' into which the conversion should be performed and generates an optimal $\boldsymbol{z}_{\text{sem.optimal}}$ which includes the semantics of a COR T1 sequence. Subsequently, the $\boldsymbol{z}_{\text{sem.original}}$ is modified by the optimal $\boldsymbol{z}_{\text{sem.optimal}}$ which is still additionally weighted by a weight factor $\kappa \in [0.4, 0.6]$ and $\sqrt{512}$ (the length of $\boldsymbol{z}_{\text{sem}}$). This process preserves the semantics of the original image COR PD but converts the features defining the MRI sequence characteristics of the COR T1 image. The linear shifting is shown in Equation 5.1.

$$\boldsymbol{z}_{\text{sem.converted}} := \boldsymbol{z}_{\text{sem.original}} + \sqrt{512}\kappa\boldsymbol{z}_{\text{sem.optimal}} \tag{5.1}$$

5.3 Edge Optimization

But a fundamental problem is encountered after the initial sequence conversions with the classifier. Instead of just changing the contrast, the conversion in latent space causes the MRI images to change in their anatomical structure. The edges of muscle tissue, bone, and ligaments are affected by the weighting of the classifier. To counteract this, methods to force the retention of these anatomical structures are considered. One method uses an a posteriori approach. This means that the structure of the Diffusion Autoencoder model is retained and only the conversion of the latent vector $\boldsymbol{z}_{\text{sem}}$ is influenced.

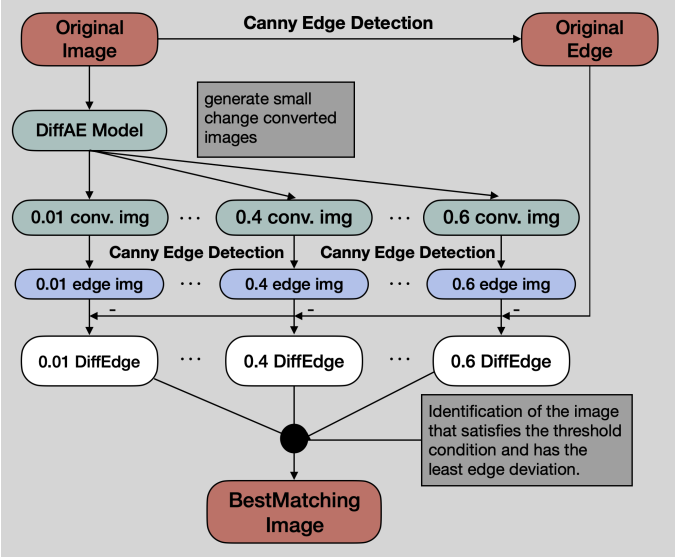


Figure 5.3: Procedure to determine the best match between the edges of the original and converted MRI image using the Canny Edge detection algorithm. First, the algorithm is applied to the original image and a variety of small-step converted images generated by weighting the conversion with the weight factor $\kappa \in (0, 0.6]$. The difference in absolute value between the original edge image and each of the manipulated edge images is computed. The best match is classified as the conversion which fulfill the threshold of 0.4 and has the least edge deviation.

The converted images correspond to an iteration of $\boldsymbol{z}_{\text{sem.original}}$ from the original image to the $\boldsymbol{z}_{\text{sem.converted}}$ corresponding to the classification of the contrast to be converted. With the threshold parameter 0.4 it is ensured, in addition to the condition of generating the edges as equally as possible, that a sequence conversion is performed. The evaluation is done by a simple summation, determining which of the pixels of the edge-difference images are not equal to zero. This corresponds to a deviation of the edges between original and converted image. The converted image that satisfies the threshold condition and has the smallest edge deviation is then identified as the best matching image. Algorithm 1 shows in detail how this method is implemented and Figure 5.3 shows the corresponding block diagram.

Algorithm 1 Procedure for determining the best match between the edges of the original and converted MRI image using the Canny Edge detection algorithm

procedure MANIPULATELATENTVECTOR
    for j in range(60) **do**
        $z_{\text{sem.smallChange}} \leftarrow z_{\text{sem}}$ shifted by stepsize, directed from classifier
        prediction $\leftarrow$ classifier forwards $z_{\text{sem.smallChange}}$
        smallChangeImage $\leftarrow$ rendering and conditioned by $z_{\text{sem.smallChange}}$
        stepsize $\leftarrow +0.01$
    end for
    return list of *smallChangeImages*, list of used *stepsizes*
end procedure

procedure EDGECLASSIFIER(*smallChangeImages*)
    originalEdgeImage $\leftarrow$ Canny Edge detection
    for k in range(len(*smallChangeImages*) **do**
        smallChangeEdgeImage $\leftarrow$ Canny Edge detection
        diffEdgeImage $\leftarrow |\, originalEdgeImage - smallChangeEdgeImage \,|$
    end for
    return list of *diffEdgeImages*
end procedure

procedure DETERMINELOWESTEDGEDEVIATION
    stepsizeThreshold $\leftarrow 0.4$
    for *diffEdgeImage* in list **do**
        for *pixel* in *diffEdgeImage* **do**
            if pixel $\neq 0$ **then**
                pixelNotEqualZero $\leftarrow +1$
            end if
        end for
        sortedDiffEdgeImage $\leftarrow$ ascending pixel not equal zero
    end for
    for *correspondingStepsize* in *sortedPixelNotEqualZero* **do**
        if *matchingStepsize* $>$ *stepsizeThreshold* **then**
            break
        end if
    end for
    return bestMatchingImage
end procedure

5.4 Metrics

The Diffusion Autoencoder is evaluated using various metrics commonly used in computer vision and deep learning model evaluation. In total, six different methods, such as the Fréchet Inception Distance (FID) score, the Precision and Recall score, the Structural Similarity (SSIM) score, the Mean Squared Error (MSE) score and the Peak Signal to Noise Ratio (PSNR) score.

Fréchet Inception Distance (FID)

The Fréchet Inception Distance (FID) score is a metric to evaluate the quality of generated images of Generative AI models. It is based on the calculation of the Fréchet distance between the distributions of the real images e.g. 5000 original COR PD images and the generated images e.g. 5000 generated COR PD images from original COR T1 images. A low FID score indicates high similarity, whereas a high FID score indicates that the generated images differ significantly from the real images [32]. To calculate the Fréchet distance between two multivariate Gaussian distributions $X_1 \sim \mathcal{N}(\mu_{X_1}, \sum_{X_1})$, $X_2 \sim \mathcal{N}(\mu_{X_2}, \sum_{X_2})$, the sample mean μ_1 is used for generated samples, the sample mean μ_2 which is precalculated on a representative dataset, the covariance matrice $\sum_{X_1}$ for generated samples and the covariance matrice $\sum_{X_2}$ which is also precalculated on the representative dataset. This allows us to define the FID score as the following:

$$FID := \|\mu_{X_1} - \mu_{X_2}\|^2 + tr\left(\sum_{X_1} + \sum_{X_2} - 2\sqrt{\sum_{X_1} * \sum_{X_2}}\right) \tag{5.2}$$

Precision and Recall

The Precision and Recall score was introduced by Kynkäänniemi et al. [33]. Precision indicates how well a generated image e.g. conversion from COR T1 to COR PD qualitatively matches the real images e.g. COR PD. Recall indicates how variant the images generated by the diffusion model are. Figure 5.4 shows the distributions P_r of the real images in blue and P_g of the generated images in red.

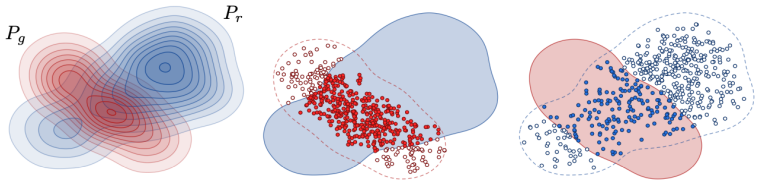


Figure 5.4: Left: P_r and P_g are the distributions of real and generated images. Both distributions represent the same sequence e.g COR PD.
Middle: Calculate whether P_r covers a random image from P_g.
Right:Calculate whether P_g covers a random image from P_r [33].

The Precision score and the Recall score are probabilities that a randomly selected generated image from P_g lies within the distribution of P_r (Precision) and that a randomly selected real image P_r lies within the distribution of P_g (Recall) [33]. The scores take values from 0.0 to 1.0. A high Precision value indicates a good image quality, while a low value indicates a low quality, which means fewer images fit into the distribution of real images. A high Recall value indicates a large variation in the generated images, while a low value indicates that the distribution of generated images has a low probability of covering the real image.

Structural Similarity (SSIM)

The SSIM score quantifies the similarity between the reference image and the generated image. This metric is based on the assumption that human perceptual characteristics such as contrast sensitivity, brightness perception, and structural understanding are important in assessing perceived image quality. A high SSIM score (close to 1) indicates high structural similarity between images, whereas a low SSIM score (close to 0) indicates low structural similarity. The score is calculated over windows of size $N \times N$ in the image. Two different windows X and Y are taken and the SSIM score of these is calculated with the following equation, for further details of the components see the paper by Zhou Wang et al. [34].

$$\text{SSIM} := \frac{(2\mu_x\mu_y + C_1)(2\sigma_{xy} + C_2)}{(\mu_x^2 + \mu_y^2 + C_1)(\sigma_x^2 + \sigma_y^2 + C_2)} \tag{5.3}$$

Peak Signal to Noise Ratio (PSNR) and MSE

The Peak Signal-to-Noise Ratio (PSNR) [35] is based on the comparison of the maximum possible signal level with the noise in the image. A higher PSNR indicates better image quality, while a lower PSNR indicates greater distortion or noise. The MSE is calculated between the pixel values of the original image X and the generated image Y, which have height h and width w respectively.

$$\text{MSE} := \frac{1}{hw} \sum_{i=0}^{h-1} \sum_{j=0}^{w-1} \big(X(i,j) - Y(i,j)\big)^2 \tag{5.4}$$

To calculate the PSNR, the value for the peak signal (MAX = 1) and the MSE is needed. The PSNR is then calculated in decibels using the following equation:

$$\text{PSNR} := 20 * \log_{10}\Big(\frac{\text{MAX}}{\sqrt{\text{MSE}}}\Big) \tag{5.5}$$

6 Evaluation and Results

In the following Section, the Diffusion Autoencoder itself is examined, then the latent sampling and interpolation ability is evaluated and finally the MRI sequence conversion is analysed by adding the classifier to the pretrained Diffusion Autoencoder. The MRI sequence conversion is rated metrically and by a radiologist. Figure 6.1 illustrates the denoising process during the decoding.

6.1 Diffusion Autoencoder

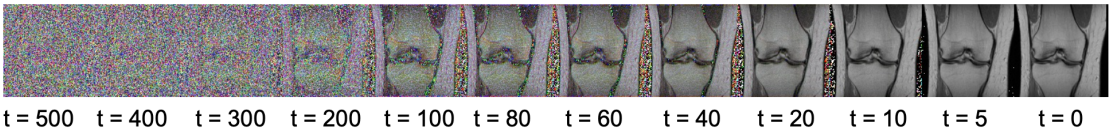


Figure 6.1: Denoising process $p_\theta(\boldsymbol{x}_{t-1}|\boldsymbol{x}_t)$ conditioned by semantic subcode $\boldsymbol{z}_{\text{sem}}$ and stochastic subcode $\boldsymbol{x}_T$. $\boldsymbol{x}_{500}$ on the left and the encoded image $\boldsymbol{x}_0$ on the right. The RGB noise is due to the three channel architecture, which is explained in Section 5.1.

The training loss function (L2-loss) of the Diffusion Autoencoder converges around a value between 0.02 and 0.03, which is plotted in Figure 6.2. To monitor the training, the L2-loss is not just calculated, but the FID score is evaluated in Figure 6.3, and the PSNR score and the MSE score are evaluated in Figure 6.4 and Figure 6.5 at intervals of 4.000.000 samples. These are calculated on a validation dataset of 10.000 images.

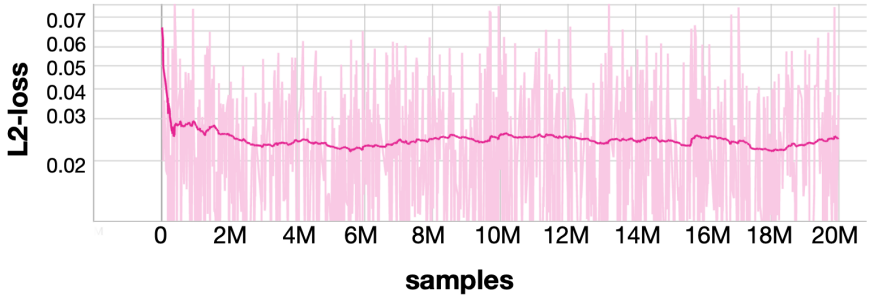
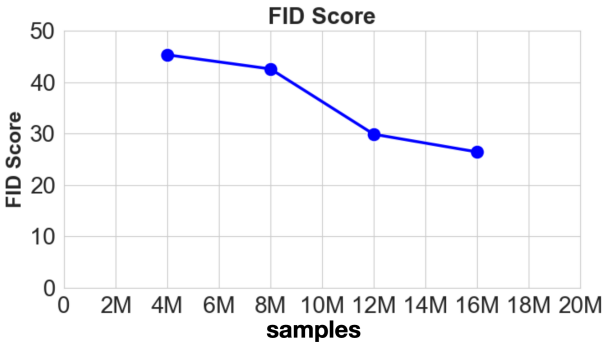


Figure 6.2: The L2-loss of the DiffAE training converges around a value between 0.02 and 0.03 over a training duration of 20.000.000 samples. The thick line is the smoothed value and the transparent line is the actual value.

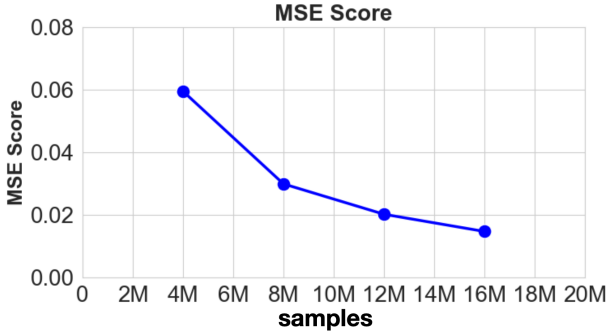


(a) Plot

Samples	FID Score ↓
04 mio.	45.3
08 mio.	42.56
12 mio.	29.87
16 mio.	26.43

(b) Table

Figure 6.3: FID score on 10.000 samples of the MRI Knee COR dataset at 4M, 8M, 12M and 16M entered samples. The FID score decreases as the number of samples increases.

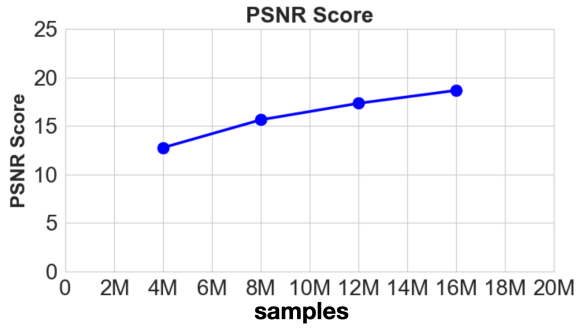


(a) Plot

Samples	MSE Score ↓
04 mio.	0.0595
08 mio.	0.0299
12 mio.	0.0202
16 mio.	0.0147

(b) Table

Figure 6.4: The MSE score on 10.000 samples of the MRI Knee COR dataset at 4M, 8M, 12M and 16M entered samples improves significantly.



(a) Plot

Samples	PSNR Score ↑
04 mio.	12.76
08 mio.	15.63
12 mio.	17.33
16 mio.	18.66

(b) Table

Figure 6.5: PSNR score on 10.000 samples of the MRI Knee COR dataset at 4M, 8M, 12M and 16M entered samples improves by analogy of the MSE score.

Encoding a test image into latent space works and so does the subsequent decoding, as shown in Figure 6.6. The SSIM score is used to calculate the structural similarity between the original and decoded images. The following example generates an SSIM score of 0.9901. Furthermore, the pixel-wise difference between the original and decoded images in absolute value is calculated and plotted in a heat map, see Figure 6.7. In the Appendix A are more test images that are encoded and subsequently decoded.

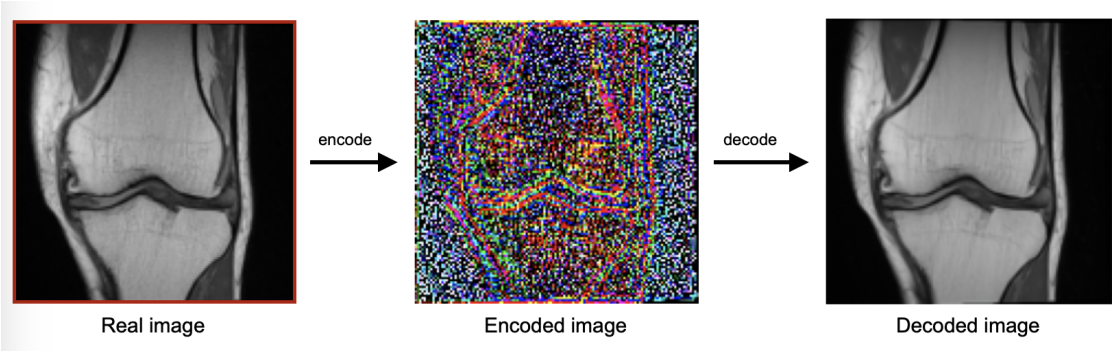


Figure 6.6: The real image $\boldsymbol{x}_0$ (left) is encoded in the latent space through the forward diffusion process $q(\boldsymbol{x}_t|\boldsymbol{x}_{t-1})$. The encoded image is a representation of $\boldsymbol{x}_T$ (middle) and subsequently the decoding through the reverse diffusion process $p_\theta(\boldsymbol{x}_{t-1}|\boldsymbol{x}_t, \boldsymbol{z}_{\text{sem}})$ to $\boldsymbol{x}_{0.\text{decoded}}$ (right) is performed. The RGB noise is due to the three channel architecture, which is explained in Section 5.1.

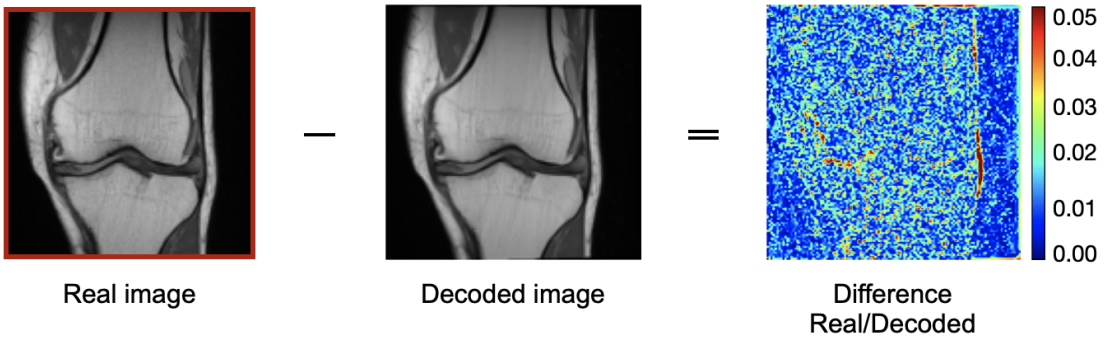


Figure 6.7: The pixelwise difference between the real image $\boldsymbol{x}_0$ (left) and the decoded image $\boldsymbol{x}_{0.\text{decoded}}$ (middle) is done by an absolute subtraction of both images. The deviations are minimal in the heatmap (right), the scale from blue to red covers a deviation of 0.00 up to 0.05. The edges of the tissue and the bones show more deviations (colored yellow up to red) than the homogenous areas (colored blue up to cyan).

Latent sampling and interpolation

Before the sequence conversion is presented in combination with the classifier, two more features of the model are introduced. First, image generation, which is related to one of the main topics in medical imaging and deep learning [36]. Second, the interpolation between two images. The success of deep learning models depends on the size and quality of the training data [28]. Since the acquisition and provision of medical data is regulated by data privacy laws, data sharing between research institutions is complex. Latent sampling is supposed to remedy the data problem by artificially synthesizing the required datasets and thus not attributing them to a real person. However, it should be added that for latent sampling, datasets must be available beforehand, because at first the constituted latent space needs to be calculated. Figure 6.8 shows a batch of 32 sampled MRI images from the latent space. To rate the quality of the these images, the FID score is calculated for the latent samples on a validation dataset of 10.000 images. The results as well as the training parameters are shown in Figure 6.10 and Table 6.1.

Table 6.1: Network architecture and training parameters of the latent training for the MRI Knee COR latent training dataset with the same 35.635 images. The final training took 48 hours on one NVIDIA Tesla Volta V100 GPUs with 100.000.000 entered samples and a batch size of 256. Other parameters are related to the used loss and optimizer

Parameter	**MRI Knee COR latent**
Batch size	256
$\mathbf{z}_{\text{sem}}$ trained	100.000.000
Total train dataset	35.635
$\mathbf{z}_{\text{sem}}$ size	512
β scheduler	constant 0.008
Learning rate	1e-4
Optimizer	AdamW (weight decay = 0.01)
Training T	1000
Diffusion loss	L1 loss with noise prediction ϵ
Num. GPUs	1
Absolute training time	$\sim$ 48h

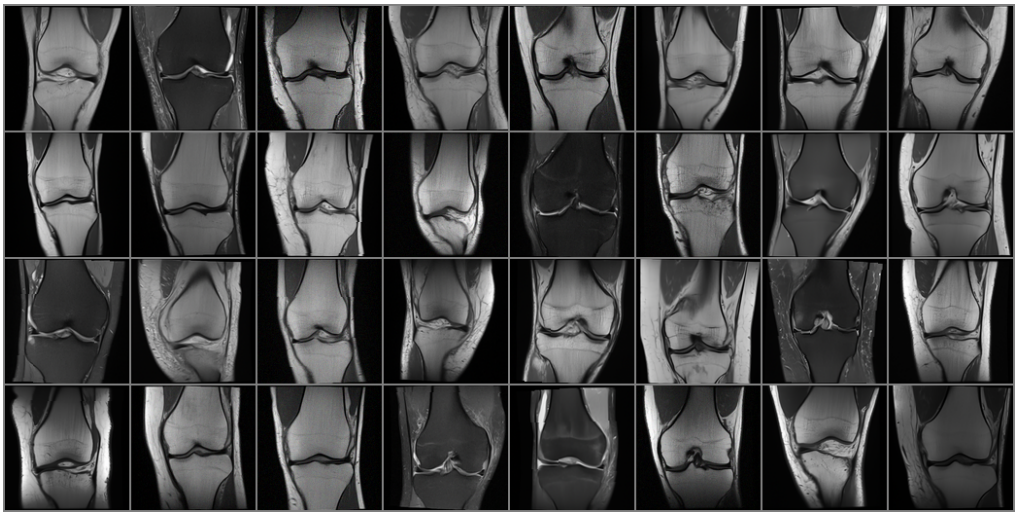


Figure 6.8: A batch of 32 latent sampled images from $\boldsymbol{x}_T \sim \mathcal{N}(0, \boldsymbol{I})$ conditioned by $\boldsymbol{z}_{\text{sem}}$.

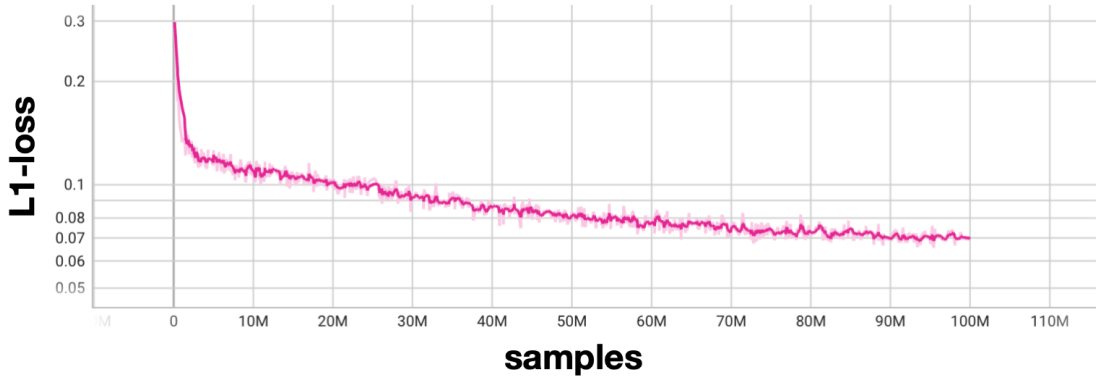
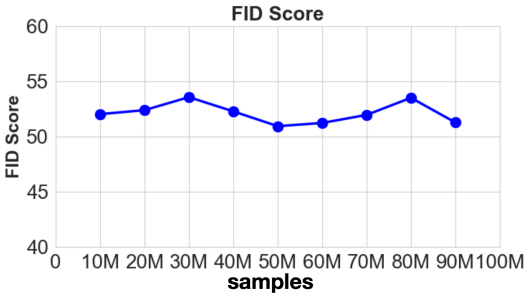


Figure 6.9: The L1-loss of the MRI Knee COR latent training converges over around a value between 0.06 and 0.08 over a training time of 100.000.000 samples or 48h. The thick line is the smoothed value and the transparent line is the actual value.



(a) Plot

Samples	FID Score ↓
10 mio.	52.03
20 mio.	52.39
30 mio.	53.57
40 mio.	52.28
50 mio.	50.94
60 mio.	51.23
70 mio.	51.96
80 mio.	53.51
90 mio.	51.27

(b) Table

Figure 6.10: The FID score calculated on 10.000 samples of the latent training does not improve from the first evaluated timestamp at 10.000.000 samples until the end of the training.

Furthermore, it is possible to interpolate between two latent vectors $\boldsymbol{z}_{\text{sem}}$ with the Diffusion Autoencoder. For this, two different MRI images are encoded into the latent space and then linearly interpolated according to Equation 6.1. The interpolation shows a preview of what sequence conversion might look like later on. The difference between this and a usable sequence conversion is that the anatomical structures of the knee are completely changed. Figure 6.11 shows such an interpolation between a COR PD sequence and a COR PD FS sequence.

$$\text{Interpolation} := \boldsymbol{z}_{\text{sem1}}(1-x) + \boldsymbol{z}_{\text{sem2}}x \tag{6.1}$$

whereas $x \in [0, 1]$. x is raised like a linear scheduler from 0 to 1 in steps of $\frac{1}{10}$. If x is closer to 0, $\boldsymbol{z}_{\text{sem1}}$ is weighted more, and if x is closer to 1, $\boldsymbol{z}_{\text{sem2}}$ is weighted more.

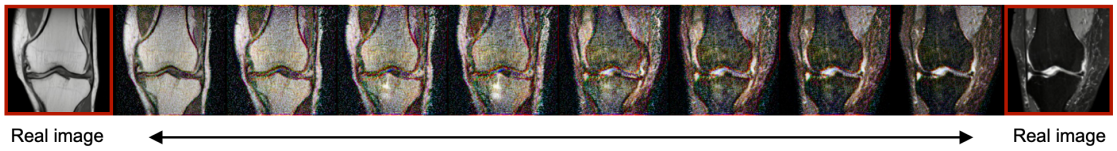
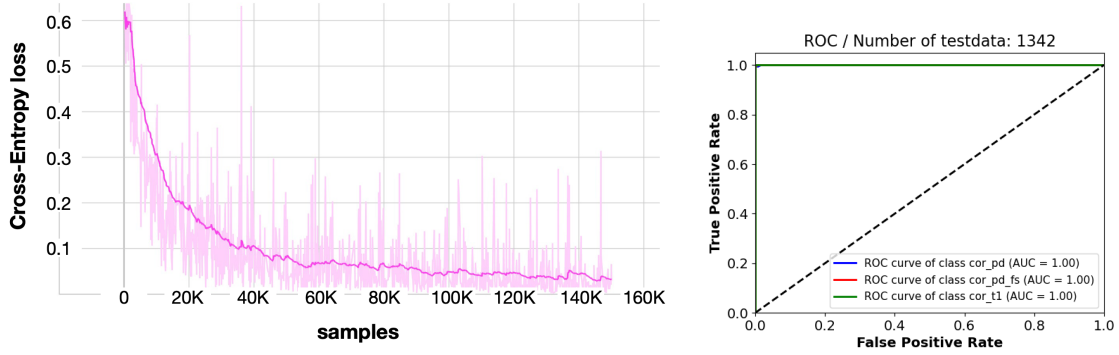


Figure 6.11: Interpolation between the two latent vectors $\boldsymbol{z}_{\text{sem1}}$ and $\boldsymbol{z}_{\text{sem2}}$. COR PD sequence on the left and COR PD FS sequence on the right. The RGB noise is due to the three channel architecture, which is explained in Section 5.1.

6.2 Classifier



(a) **(b)**

Figure 6.12: **a)** Classifier training. The Cross-Entropy loss converges to a value between 0.03 and 0.05 over 150.000 entered samples
b) Classifier test. The ROC curve is the same for all three classes COR PD, COR PD FS and COR T1. It is a 100% true positive rate, a perfect classifier.

The Cross-Entropy loss of the classifier converges to a value between 0.03 and 0.05, which is shown in Figure 6.12(a). It achieves an AUC of 1.0 for all three

classes despite using augmentations, which corresponds to a perfect classifier, see Figure 6.12(b). The confusion matrix in Figure 6.13 shows that only one up to two MRI sequences are incorrectly classified.

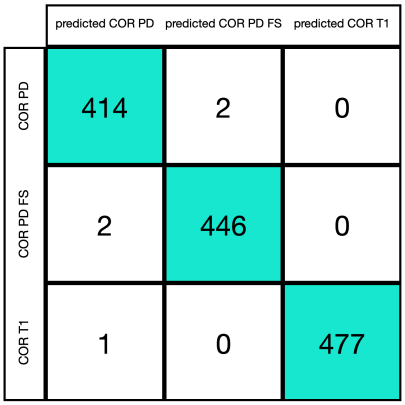


Figure 6.13: Classifier test with 1342 test samples. The confusion matrix shows in the columns the predicted class and in the rows the actual class. 414 times, 446 times and 477 times the MRI sequences are correctly classified. Only 1 COR T1 sequence is mistaken with a COR PD classification and 2 COR PD sequences are mistaken as COR PD FS sequences and vice versa.

6.3 Sequence Conversion

An evaluation of the sequence-converted MRI images is performed by a radiologist of the University Hospital Aachen and the FID score and the Precision and Recall score are calculated. The radiological evaluation, coupled with consideration of the MRI signal quality correspondence and the anatomical structure correspondence, is made by a radiologist using an evaluation scheme of subjective perception. Moreover, Figure 6.14 shows a detailed progress on how the sequence conversion for the six different conversions is performed.

Metric evaluation

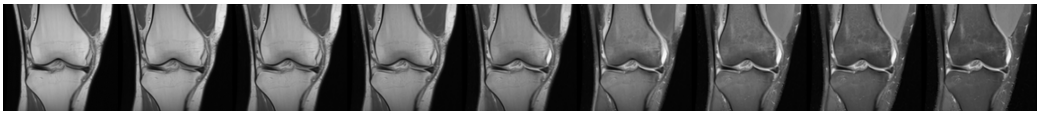
Each of the other two sequences are synthesized from each original sequence. One dataset of a synthesized sequence contains of 5000 samples. However, due to resource capacity, the Edge Optimization algorithm is not used for the synthesis of the 5000 samples. Therefore, a lower and upper bound of the weight factor κ are calculated empirically for the shift of the latent vector $\boldsymbol{z}_{\text{sem}}$. The lower bound corresponds to a value of $\kappa_{\text{lower}} = 0.4$ and the upper bound corresponds to a value of $\kappa_{\text{upper}} = 0.6$. Accordingly, not only two datasets are generated from an original sequence, but four, each containing the same synthesized sequence twice but with a different weight factor applied. Table 6.2 provides an overview of the individual datasets that are compared. Furthermore, the FID scores and the Precision and

Recall values, which are calculated according to the presented metrics from Section 5.4, are tabulated there.

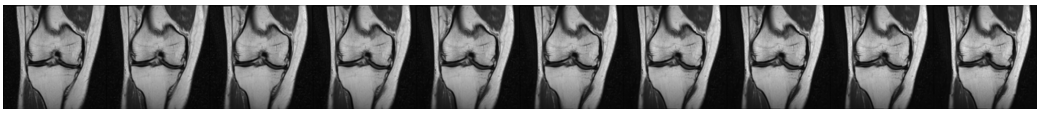
Table 6.2: Sequence conversion analysis with FID score and Precision and Recall score. Each MRI Sequence is converted by a weight factor $\kappa = 0.4$ and $\kappa = 0.6$ in the other two sequences and vice versa

Sequence	**generated Seq.**	**weight factor κ**	**FID score↓**	**Precision↑**	**Recall↑**
COR PD	COR PD FS	0.4	37.3	0.19	0.24
	COR PD FS	0.6	26.2	0.26	0.23
	COR T1	0.4	28.1	0.07	0.41
	COR T1	0.6	22.8	0.12	0.42
COR PD FS	COR PD	0.4	31.0	0.11	0.35
	COR PD	0.6	33.0	0.10	0.28
	COR T1	0.4	104.2	0.01	0.25
	COR T1	0.6	104.2	0.01	0.25
COR T1	COR PD	0.4	49.2	0.07	0.1
	COR PD	0.6	65.1	0.03	0.05
	COR PD FS	0.4	96.2	0.06	0.04
	COR PD FS	0.6	50.1	0.14	0.04

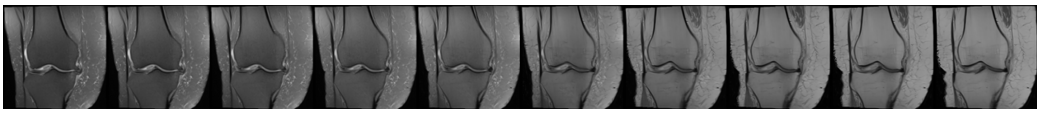
The sequence conversion from COR PD to COR PD FS and COR T1 achieves the lowest FID score (COR PD FS: 37.3 ($\kappa = 0.4$) and 26.2 ($\kappa = 0.6$))(COR T1: 28.1 ($\kappa = 0.4$) and 22.8 ($\kappa = 0.6$)) in relation to the conversion COR PD FS to COR PD and COR T1 and the conversion COR T1 to COR PD and COR PD FS. A higher weight factor $\kappa = 0.6$ in relation to $\kappa = 0.4$ leads to a decreasing FID score for the conversion COR PD to COR PD FS and COR T1. The sequence conversion from COR PD FS to COR PD performs similarly according to the FID score (31.0 ($\kappa = 0.4$) and 33.0 ($\kappa = 0.6$)) but the conversion from COR PD FS to COR T1 has the highest FID score of 104.2, indicating that the sampled images do not sufficiently match the real images. The sequence conversion from COR T1 to the COR PD and COR PD FS sequences performs worse than the first mentioned COR PD conversion according to the FID score (COR PD: 49.2 ($\kappa = 0.4$) and 65.1 ($\kappa = 0.6$))(COR PD FS: 96.1 ($\kappa = 0.4$) and 50.1 ($\kappa = 0.6$)) but better than the COR PD FS conversion.



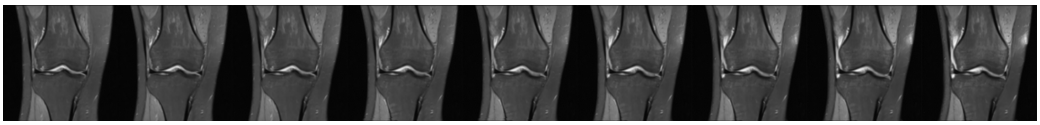
(a) COR PD to COR PD FS



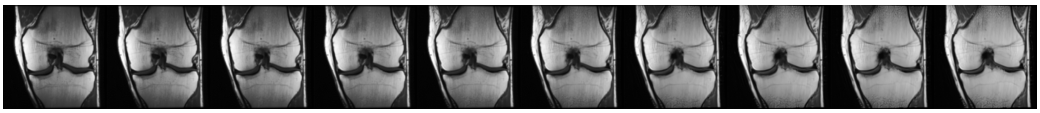
(b) COR PD to COR T1



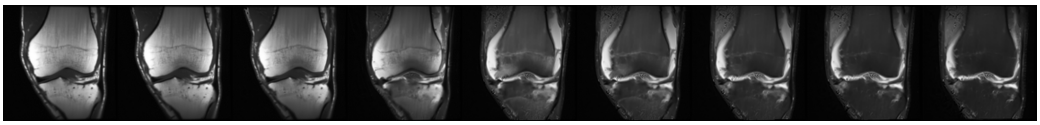
(c) COR PD FS to COR PD



(d) COR PD FS to COR T1



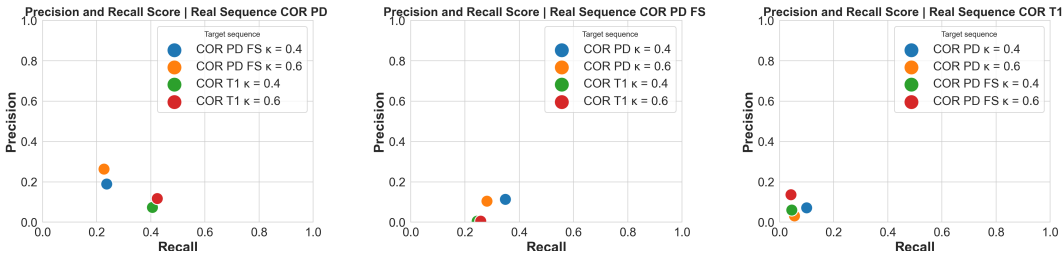
(e) COR T1 to COR PD



(f) COR T1 to COR PD FS

Figure 6.14: Sequence conversion results with Edge Optimization algorithm. From the real image on the left, a gradually shifting of the latent vector $\boldsymbol{z}_{\text{sem}}$ towards the target MRI sequence on the right is visualized.

The Precision and Recall evaluation of the COR PD to COR PD FS conversion, as shown in Figure 6.15(a), results in a higher Precision score in comparison to the COR PD to COR T1 conversion (COR PD FS: 0.19 ($\kappa = 0.4$), COR T1: 0.07 ($\kappa = 0.4$) and COR PD FS: 0.26 ($\kappa = 0.6$), COR T1: 0.12 ($\kappa = 0.6$)). A higher weight factor $\kappa = 0.6$ in relation to $\kappa = 0.4$ leads to an increasing Precision score for these conversions. The Recall score is higher for the COR PD to COR PD FS conversion is lower in comparison to the COR PD to COR T1 conversion (COR PD FS: 0.24 ($\kappa = 0.4$), COR T1: 0.41 ($\kappa = 0.4$) and COR PD FS: 0.23 ($\kappa = 0.6$), COR T1: 0.42 ($\kappa = 0.6$)). The Precision scores for the converted COR T1 and COR PD sequences are below 0.2 (Figure 6.15(b)). The results are similar for the conversion from the real sequence COR T1. Figure 6.15(c) shows that Precision and Recall scores are smaller than 0.2 for both converted sequences.

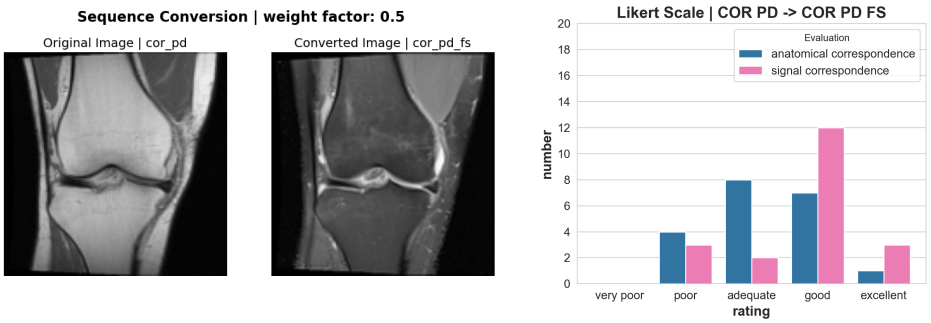


(a) Real Seq. COR PD **(b)** Real Seq. COR PD FS **(c)** Real Seq. COR T1

Figure 6.15: Precision and Recall scores for all conversions from Table 6.2.

Radiological evaluation

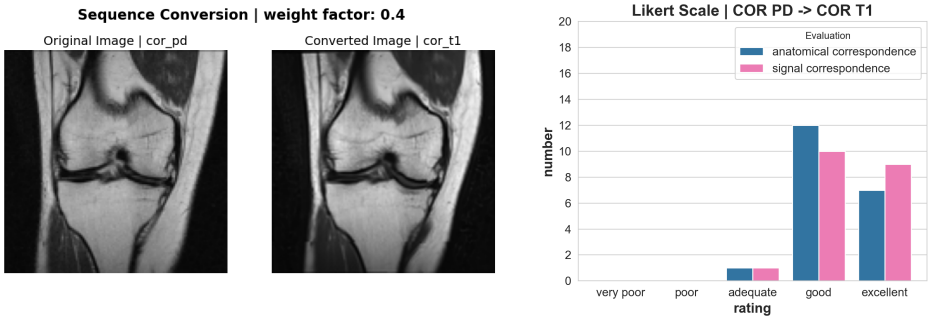
In addition to the FID score and the Precision and Recall metric, an evaluation scheme for the radiological evaluation is developed. The evaluation is made by Robert Siepmann, assistant physician at University Hospital Aachen. MRI images are generated by the Diffusion Autoencoder with addition of the Edge Optimization algorithm. In the first part, the radiologist rates the anatomical correspondence, such as bone contours and tissue between the original image and the generated image, and in the second part, the radiologist rates the signal correspondence. These two categories are rated according to the Likert scale in ascending order from 'very poor', 'poor', 'adequate' to 'good' and 'excellent'. The Diffusion Autoencoder randomly selects twenty original images of each sequence from the test dataset, which are then converted to the remaining two sequences. Thus, the radiologist evaluates a total of 120 sequence conversions, divided into 6 different conversion directions, each containing $N = 20$ images. The following are the Likert Scale evaluations of the radiologist paired with an example sequence conversion and the calculated weight factor κ. The other evaluated images can be found in the Appendix A.2.



(a) Seq. conversion COR PD to COR PD FS **(b)** Likert scale evaluation $N = 20$

Figure 6.16: The signal properties are rated as poor to excellent and the preservation of the anatomical structures is rated as poor to excellent as well.

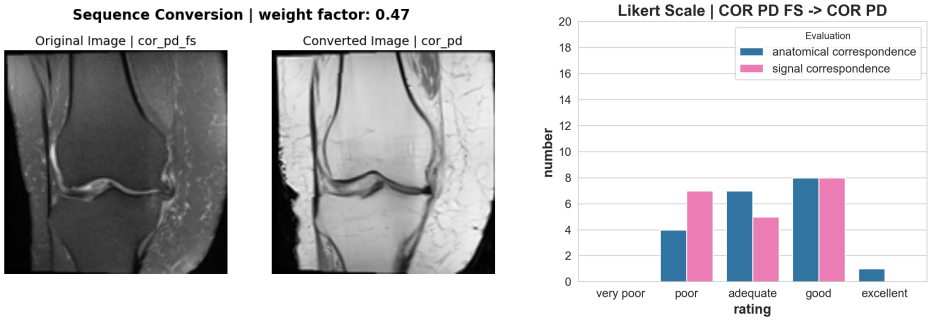
According to the radiologist, the sequence conversion from COR PD to COR PD FS is rated as poor to excellent in terms of signal quality (pink) and the preservation of bone and tissue structures is rated similarly as poor to excellent (blue), as can be seen in Figure 6.16. It is also worth to look at the Edge Optimized weight factor κ. It does not follow a pattern in this conversion and lies between the lower bound $\kappa_{\text{lower}} = 0.4$ and the upper bound $\kappa_{\text{upper}} = 0.6$. These values can be traced in the Appendix A.2.



(a) Seq. conversion COR PD to COR T1 **(b)** Likert scale evaluation $N = 20$

Figure 6.17: The signal properties are rated adequate up to excellent converted and the preservation of the anatomical structures is rated similarly.

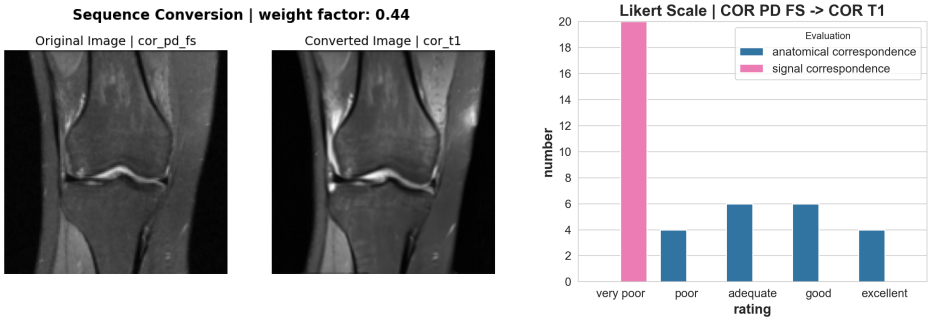
In contrast, the sequence conversion from COR PD to COR T1 in Figure 6.17 is rated better than the conversion from COR PD to COR PD FS. Both anatomically and in terms of signal quality, the radiologist rates this conversion as adequate up to excellent. Looking at the Appendix A.4, the Edge Optimization algorithm classifies the lower bound $\kappa_{\text{lower}} = 0.4$ as the best match of the edges for all 20 test images.



(a) Seq. conversion COR PD FS to COR PD **(b)** Likert scale evaluation $N = 20$

Figure 6.18: The signal properties of the converted image are rated as poor to good and the preservation of the anatomical structures is rated as poo up to excellent.

The sequence conversion from the COR PD FS sequence to the COR PD sequence, see Figure 6.18, is rated 7 times poor, 5 times adequate and 8 times good in terms of signal quality (pink) which is a lower rating than the conversion vice versa (Figure 6.16). The anatomical correspondence is equally mediocre (blue). Again, the weight factor κ takes values between the lower and upper bound.

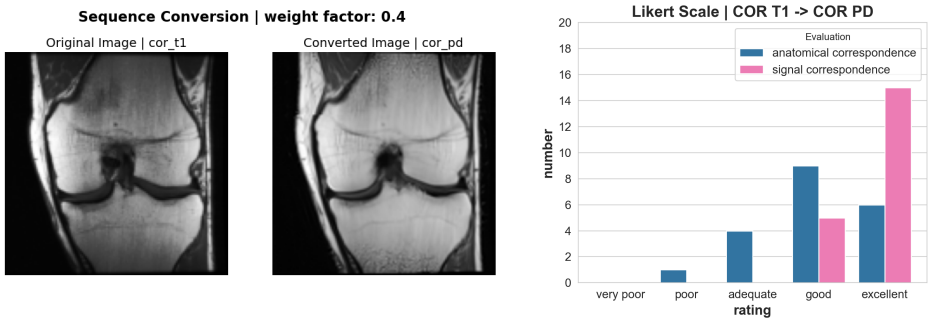


(a) Seq. conversion COR PD FS to COR T1

(b) Likert scale evaluation $N = 20$

Figure 6.19: The signal properties are rated very poor converted but the preservation of the anatomical structures is rated between poor and excellent.

The signal correspondence of the COR PD FS to COR T1 conversion is evaluated as very poor (pink) and as can be seen in Figure 6.19(a), the conversion does not work. Despite that, the anatomical conversion is rated in between poor and excellent (blue). The exact reason is not obvious, since the classifier works perfectly according to the tests (Section 6.2) and as shown in the following, the conversion works in the opposite way (Figure 6.21).



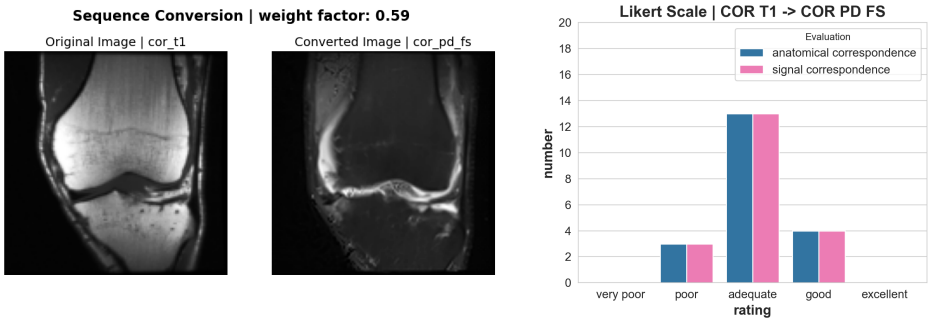
(a) Seq. conversion COR T1 to COR PD

(b) Likert scale evaluation $N = 20$

Figure 6.20: The signal properties of the converted image are rated excellent and good and the preservation of the anatomical structures is rated as poor up to excellent.

The conversion of COR T1 to COR PD is presented in Figure 6.20. This is evaluated as good up to excellent in terms of signal quality (pink), and the bones

and the tissue are converted as good up to excellent (blue). As with the reverse conversion, the Edge Optimization algorithm only classifies the lower bound $\kappa = 0.4$ as the best match.



(a) Seq. conversion COR T1 to COR PD FS

(b) Likert scale evaluation $N = 20$

Figure 6.21: The signal properties and the preservation of the anatomical structures are rated similarly as poor to good.

Finally, the COR T1 sequence is converted to the COR PD FS sequence (Figure 6.21). This conversion is rated as poor to good by the radiologist, both anatomically and signal-wise. However, it is interesting to note that the conversion of COR T1 to COR PD FS works in terms of the signal quality as indicated for example by the contrast of the bone, but the reverse transformation fails.

7 Discussion and Outlook

In the following, the results and the performance of the models are discussed, compared, limiting factors are addressed and an outlook on possible approaches and ideas that could be tried in follow-up work are given.

Discussion

The classifier achieves the best possible results (AUC = 1.0) for all three classes. Comparing this classification task to others e.g. the attribute classification on the CelebA-HQ dataset (average AUC = 0.92) by the exact same linear classifier in Preechakul et al. [2] paper, it is a really good performance but not elusive. Since at least the COR PD and COR PD FS sequences can be clearly distinguished from each other by contrast and brightness, the classification task is in comparison to more complex images e.g. RGB color space or different image views, simple. More encouraging is the result that the classifier can clearly distinguish a COR PD sequence from a COR T1 sequence because they are visually similar.

Moreover, it was noticeable in the metric evaluation that no pattern in the improvement of the FID score could be identified according to the weight factor κ. A larger κ shifts the latent vector $\boldsymbol{z}_{\text{sem}}$ further towards the target sequence. The hypothesis is, that this will improve the metric score. However, this behavior is only observed in half of the conversions. Contrary to the expectations, the FID score increases with increasing κ in the experiments in which COR PD sequences are generated. But no precise conclusions can be drawn from those results because the FID score and the Precision and Recall score are not specifically developed metrics for the analysis of generated medical images but have been developed for the evaluation of Generative AI generated images.

The comparison between metric evaluation and radiological evaluation reveals that especially the sequence conversion COR PD to COR PD FS and vice versa showed a better result in the FID score (COR PD to COR PD FS: 37.3 ($\kappa = 0.4$) and 26.2 ($\kappa = 0.6$)) and (COR PD FS to COR PD: 31.0 ($\kappa = 0.4$) and 33.0 ($\kappa = 0.6$)) compared to the radiological evaluation. However, the metrics for evaluating Generative AI generated images do not exactly meet the requirements of the task, as they only stochastically compare how well the generated images match the original dataset. This means there is information about how well, for example, an artificially generated COR PD sequence corresponds to an original COR PD sequence, but not how well a conversion from a specific COR PD FS sequence to that exact corresponding COR PD sequence works. The radiologist, in turn, combines the rating of signal quality and the preservation of the anatomical structure.

The conversion COR PD to COR T1 and vice versa is rated with a low Precision score (COR PD to COR T1: 0.07 ($\kappa = 0.4$) and 0.12 ($\kappa = 0.6$)) and (COR T1 to COR PD: 0.07 ($\kappa = 0.4$) and 0.03 ($\kappa = 0.6$)), see Table 6.2, indicating a low correspondence of the generated images with the original dataset from the same sequence. However, the radiologist's evaluation shows, in contrast, that especially the conversion from COR T1 to COR PD (Figure 6.20) is rated as good up to excellent in the signal quality of the generated images. One possible reason why that conversion succeeds better than the one to COR PD FS is the fact that the COR PD sequence and the COR T1 sequence are visually similar. Since the latent vectors $\boldsymbol{z}_{\text{sem}}$ are closer together in the latent space, it may simplify the modification of the latent vector. Moreover, it may be more difficult for the radiologist to see false conversions with the human eye if the MRI sequences deviate only a little.

Why the conversion from COR PD FS to COR T1 did not work (Figure 6.19), can only be hypothesized. It contradicts the previous performance of the model that the signal properties do not change optically. The classifier was tested several times and worked without errors. One possible cause is the presumed large distance between the ideal latent vectors of the sequences, although this presumption is invalidated by the working reconversion between the two sequences (Figure 6.20).

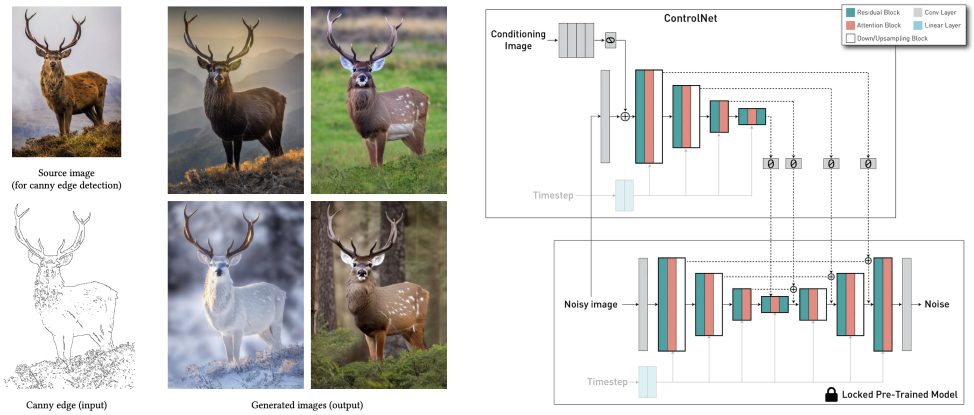
This thesis successfully shows that MRI sequence conversion is possible with a Diffusion Autoencoder. The model learns the MRI signal characteristics correctly and allows with one exception (COR PD FS to COR T1, Figure 6.19) a transformation into one of the other sequences. This thesis also revealed that the Diffusion Autoencoder alone is not sufficient for the complex task. Essential for a conversion while preserving the anatomical structures is an optimization under another constraint or by another model.

Limitations

This work has several limitations. The results of Diffusion models and in general Deep Learning models depend on the quality of the data e.g. image resolution, variety of images and number of images in the dataset [28]. Due to insufficiently available axial MRI sequences in public datasets such as MRNet, FastMRI, and OAI, the internal test dataset recorded by ourselves was not used. Instead, sufficiently available coronal sequences from the knee were used. Due to very high computer resources (2 GPUs/132h/20.000.000 samples) for training the Diffusion Autoencoder, the resolution of the images had to be downscaled. This deprives the grayscale MRI image of valuable information and makes it more difficult to evaluate using the Edge Optimization algorithm. A key limiting factor in why the radiologist's evaluation of the anatomical correspondence was mediocre is that the performed optimization was only a posteriori according to the constraint with the Edge Optimization algorithm. The current Diffusion Autoencoder based on an architecture by Preechakul et al. [2] does not provide a way during training to extend the loss of the model to multiple constraints.

Outlook

Follow-up work should examine the behavior of the Diffusion Autoencoder during training with axial MRI sequences of the lower limb. For this purpose, sufficiently quality datasets must be available. Moreover the architecture can be reduced to one channel input layers unlike the current three channel input layers. This would decrease the requirements for computational resources and the complexity of the network which, in turn, could accelerate the training and as shown in Figure 6.3 improve the FID score.



(a) ControlNet performance with Canny Edge detection [37] to generate different image contrasts while maintaining the object structure of the deer.

(b) ControlNet architecture [38] on top of a pre-trained Diffusion Model to control the latent space.

Figure 7.1: ControlNet on top of a Diffusion Model [37, 38]

In addition, the architecture of the diffusion model should be improved so that the latent space can be actively influenced, for example, to optimize the anatomical structural conversion during training by guiding specific parameters of the latent vector $\boldsymbol{z}_{\text{sem}}$ in the latent space. This is the current state of the art in artificial intelligence research and in diffusion models in particular. Current successes are achieved with so-called ControlNets, which were published in February 2023 by Zhang et al. [37]. These showed considerable success in initial trials outside the medical field. As can be seen in Figure 7.1(a), Zhang et al. also implemented those ControlNets with the option to use the Canny Edge detector to obtain the shape of the object in the image. Figure 7.1(b) shows an architecture of the ControlNet presented by Pinaya [38] in June 2023. This allows to perform the sequence conversion on a paired brain MRI images dataset. This and other publications on ControlNets show that they offer a promising way to perform clinically usable conversion between MRI sequences.

Bibliography

[1] J. Schock, D. Truhn, D. Nürnberger, S. Conrad, M. S. Huppertz, S. Keil, C. Kuhl, D. Merhof, and S. Nebelung, "Artificial intelligence-based automatic assessment of lower limb torsion on MRI," *Scientific Reports*, vol. 11, p. 23244, Dec 2021.

[2] K. Preechakul, N. Chatthee, S. Wizadwongsa, and S. Suwajanakorn, "Diffusion autoencoders: Toward a meaningful and decodable representation," in *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2022.

[3] O. Dössel, *Bildgebende Verfahren in der Medizin Von der Technik zur medizinischen Anwendung*. Berlin, Heidelberg: Springer Berlin Heidelberg, 2. aufl. 2016 ed., 2016.

[4] M. A. Bernstein, K. F. King, and X. J. Zhou, "Chapter 14 - Basic pulse sequences," in *Handbook of MRI Pulse Sequences* (M. A. Bernstein, K. F. King, and X. J. Zhou, eds.), pp. 579–647, Burlington: Academic Press, 2004.

[5] A. D. Elster, "Turbospine echo pulse and singlespine echo pulse." https://mriquestions.com/what-is-fsetse.html. [Accessed 13.07.2023].

[6] H. H. U. Düsseldorf, "T1 relaxation and T2 relaxation." https://nmr.hhu.de/asets/theory_c.html. [Accessed 19.08.2023].

[7] M. A. Bernstein, K. F. King, and X. J. Zho, "Chapter 17 - Advanced pulse sequence techniques," in *Handbook of MRI Pulse Sequences* (M. A. Bernstein, K. F. King, and X. J. Zho, eds.), pp. 802–954, Burlington: Academic Press, 2004.

[8] S. Bhattacharyya, V. Snasel, A. E. Hassanien, S. Saha, and B. K. Tripathy, eds., *Deep Learning; Research and Applications*. Berlin, Boston: De Gruyter, 2020.

[9] V. S.K., P. S.R., and V. S., *Deep Learning: A Comprehensive Guide (1st ed.)*. Boca Raton, Florida: Chapman and Hall/CRC, 2021.

[10] Phung and Rhee, "A high-accuracy model average ensemble of convolutional neural networks for classification of cloud image patches on small datasets," *Applied Sciences*, vol. 9, p. 4500, 10 2019.

[11] K. He, X. Zhang, S. Ren, and J. Sun, "Deep residual learning for image recognition," in *2016 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 770–778, 2016.

[12] J. Bridle, "Training stochastic model recognition algorithms as networks can lead to maximum mutual information estimation of parameters," in *Advances in Neural Information Processing Systems* (D. Touretzky, ed.), vol. 2, Morgan-Kaufmann, 1989.

[13] C. Sammut and G. I. Webb, eds., *Mean Squared Error*, pp. 653–653. Boston, MA: Springer US, 2010.

[14] C. Sammut and G. I. Webb, eds., *Mean Absolute Error*, pp. 652–652. Boston, MA: Springer US, 2010.

[15] A. Semenov, V. Boginski, and E. L. Pasiliao, "Neural networks with multi-dimensional cross-entropy loss functions," in *Computational Data and Social Networks* (A. Tagarelli and H. Tong, eds.), pp. 57–62, Springer International Publishing, 2019.

[16] S. Ruder, "An overview of gradient descent optimization algorithms," *CoRR*, vol. abs/1609.04747, 2016. [Online] Available: http://arxiv.org/abs/1609.04747.

[17] O. Ronneberger, P. Fischer, and T. Brox, "UNet: Convolutional networks for biomedical image segmentation," in *Medical Image Computing and Computer-Assisted Intervention – MICCAI 2015* (N. Navab, J. Hornegger, W. M. Wells, and A. F. Frangi, eds.), pp. 234–241, Springer International Publishing, 2015.

[18] J. Sohl-Dickstein, E. Weiss, N. Maheswaranathan, and S. Ganguli, "Deep unsupervised learning using nonequilibrium thermodynamics," in *Proceedings of the 32nd International Conference on Machine Learning* (F. Bach and D. Blei, eds.), vol. 37 of *Proceedings of Machine Learning Research*, (Lille, France), pp. 2256–2265, PMLR, 07–09 Jul 2015.

[19] J. Ho, A. Jain, and P. Abbeel, "Denoising diffusion probabilistic models," in *Advances in Neural Information Processing Systems* (H. Larochelle, M. Ranzato, R. Hadsell, M. Balcan, and H. Lin, eds.), vol. 33, pp. 6840–6851, Curran Associates, Inc., 2020.

[20] J. Song, C. Meng, and S. Ermon, "Denoising diffusion implicit models," *CoRR*, vol. abs/2010.02502, 2022. [Online] Available: https://doi.org/10.48550/arXiv.2010.02502.

[21] P. Dhariwal and A. Nichol, "Diffusion models beat GANs on image synthesis," *CoRR*, vol. abs/2105.05233, 2021. [Online] Available: https://doi.org/10.48550/arXiv.2105.05233.

[22] G.-X. Yuan, C.-H. Ho, and C.-J. Lin, "Recent advances of large-scale linear classification," *Proceedings of the IEEE*, vol. 100, no. 9, pp. 2584–2603, 2012.

[23] "Multiclass Classification with NumPy and TMVA; root numpy 4.7.3.dev0 documentation scikit-hep.org." http://scikit-hep.org/root_numpy/auto_examples/tmva/plot_multiclass.html. [Accessed 11.08.2023].

[24] T. Fawcett, "Introduction to ROC analysis," *Pattern Recognition Letters*, vol. 27, pp. 861–874, 06 2006.

[25] F. Schoonjans, "ROC curve analysis — medcalc.org." https://www.medcalc.org/manual/roc-curves.php. [Accessed 08.07.2023].

[26] J. Canny, "A computational approach to edge detection," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. PAMI-8, no. 6, pp. 679–698, 1986.

[27] "OpenCV: Canny Edge Detection — docs.opencv.org." https://docs.opencv.org/3.4/da/d22/tutorial_py_canny.html. [Accessed 09.07.2023].

[28] M. Z. Alom, T. Taha, C. Yakopcic, S. Westberg, P. Sidike, M. Nasrin, M. Hasan, B. Van Essen, A. Awwal, and V. Asari, "A state-of-the-art survey on deep learning theory and architectures," *Electronics*, vol. 8, p. 292, 03 2019.

[29] "MRNet: A Dataset of KneeMRs and Competition for Automated Knee MR Interpretation. — stanfordmlgroup.github.io." https://stanfordmlgroup.github.io/competitions/mrnet/. [Accessed 22.05.2023].

[30] "fastMRI Dataset — fastmri.med.nyu.edu." https://fastmri.med.nyu.edu. [Accessed 22.05.2023].

[31] "Torchvision; Torchvision 0.15 documentation — pytorch.org." https://pytorch.org/vision/stable/index.html. [Accessed 18.07.2023].

[32] M. Heusel, H. Ramsauer, T. Unterthiner, B. Nessler, and S. Hochreiter, "GANs trained by a two time-scale update rule converge to a local nash equilibrium," in *Proceedings of the 31st International Conference on Neural Information Processing Systems*, NIPS'17, (Red Hook, NY, USA), p. 6629–6640, Curran Associates Inc., 2017.

[33] T. Kynkäänniemi, T. Karras, S. Laine, J. Lehtinen, and T. Aila, "Improved precision and recall metric for assessing generative models," in *Proceedings of the 33rd International Conference on Neural Information Processing Systems*, (Red Hook, NY, USA), Curran Associates Inc., 2019.

[34] Z. Wang, A. Bovik, H. Sheikh, and E. Simoncelli, "Image quality assessment: from error visibility to structural similarity," *IEEE Transactions on Image Processing*, vol. 13, no. 4, pp. 600–612, 2004.

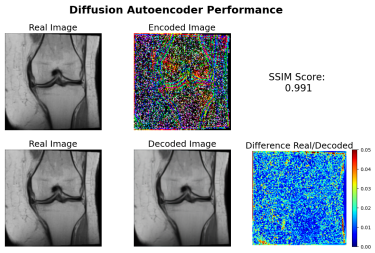
[35] A. Horé and D. Ziou, "Image quality metrics: PSNR vs. SSIM," in *2010 20th International Conference on Pattern Recognition*, pp. 2366–2369, 2010.

[36] T. Wang, Y. Lei, Y. Fu, J. F. Wynne, W. J. Curran, T. Liu, and X. Yang, "A review on medical imaging synthesis using deep learning and its clinical applications," *Journal of Applied Clinical Medical Physics*, vol. 22, no. 1, pp. 11–36, 2021.

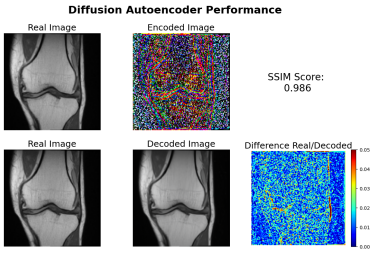
[37] L. Zhang and M. Agrawala, "Adding conditional control to text-to-image diffusion models," *CoRR*, vol. abs/2302.05543, 2023. [Online] Available: https://doi.org/10.48550/arXiv.2302.05543.

[38] W. H. L. Pinaya, "Controllable Medical Image Generation with ControlNets — towardsdatascience.com." https://towardsdatascience.com/controllable-medical-image-generation-with-controlnets-48ef33dde652. [Accessed 11.08.2023].

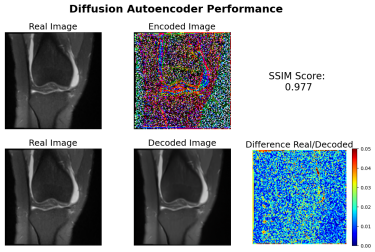
A Appendix



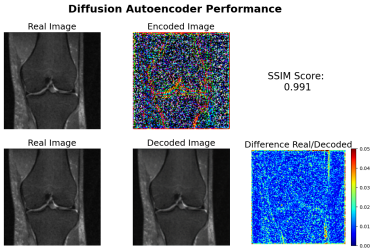
(a) COR PD real image | SSIM: 0.991↑



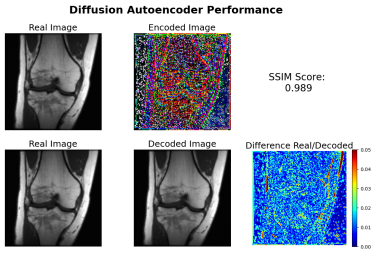
(b) COR PD real image | SSIM: 0.986↑



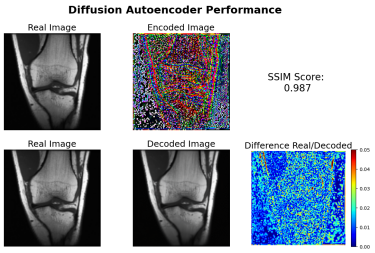
(c) COR PD FS real image | SSIM: 0.977↑



(d) COR PD FS real image | SSIM: 0.991↑

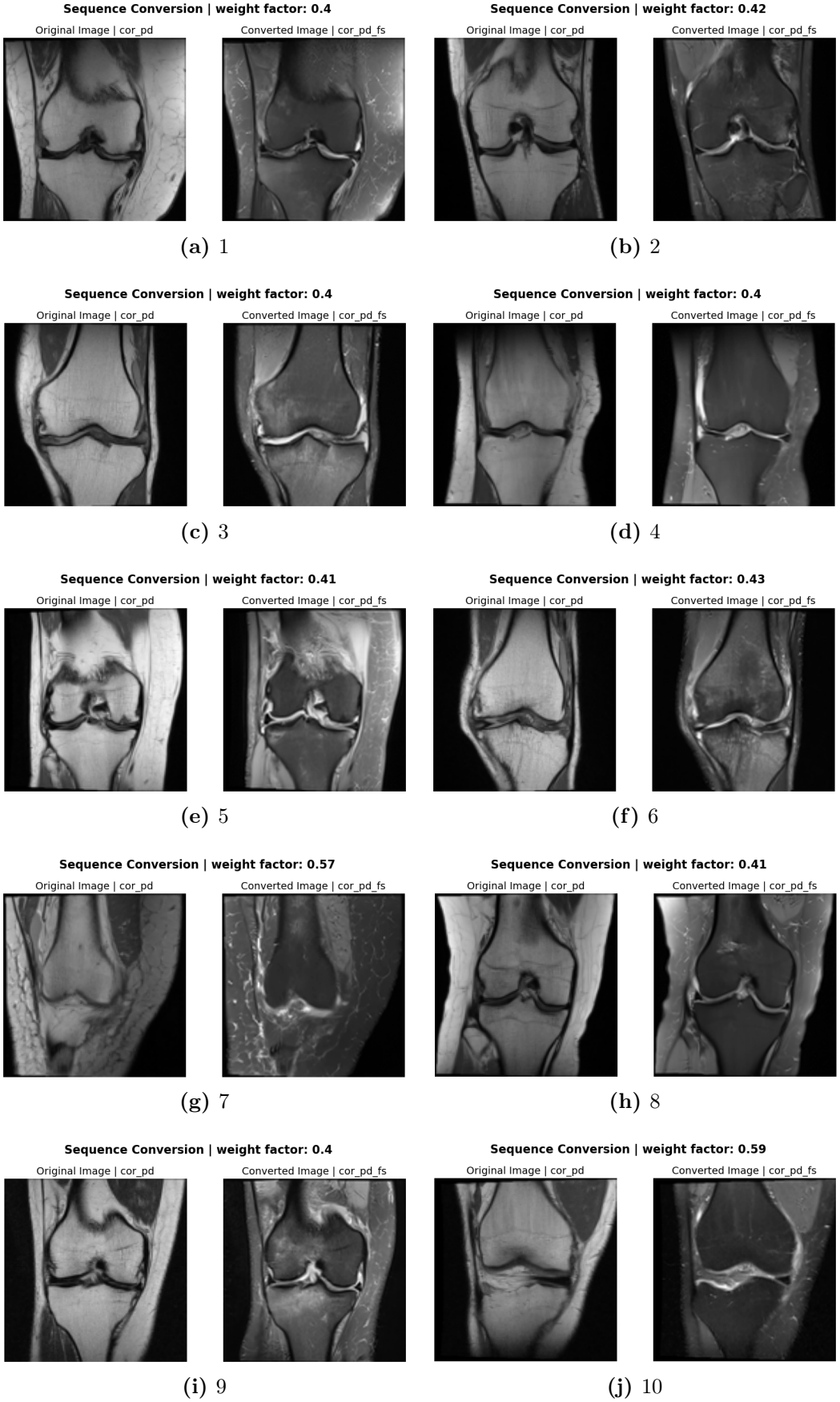


(e) COR PD T1 real image | SSIM: 0.989↑



(f) COR PD T1 real image | SSIM: 0.987↑

Figure A.1: Diffusion Autoencoder: Encoding and Decoding & SSIM score and pixelwise difference



(a) 1 **(b)** 2

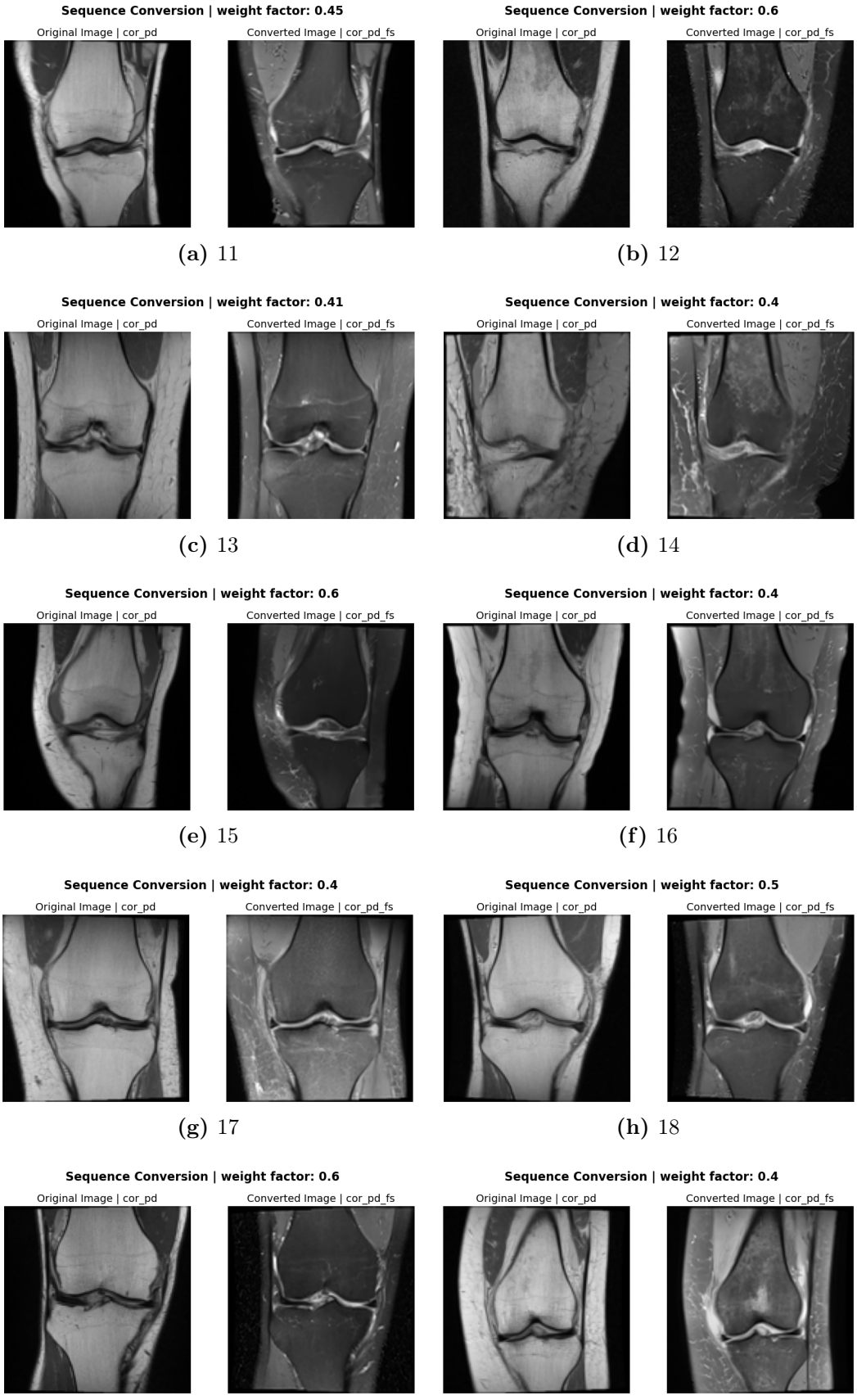
(c) 3 **(d)** 4

(e) 5 **(f)** 6

(g) 7 **(h)** 8

(i) 9 **(j)** 10

Figure A.2: Sequence Conversion COR PD to COR PD FS



(a) 11 **(b)** 12

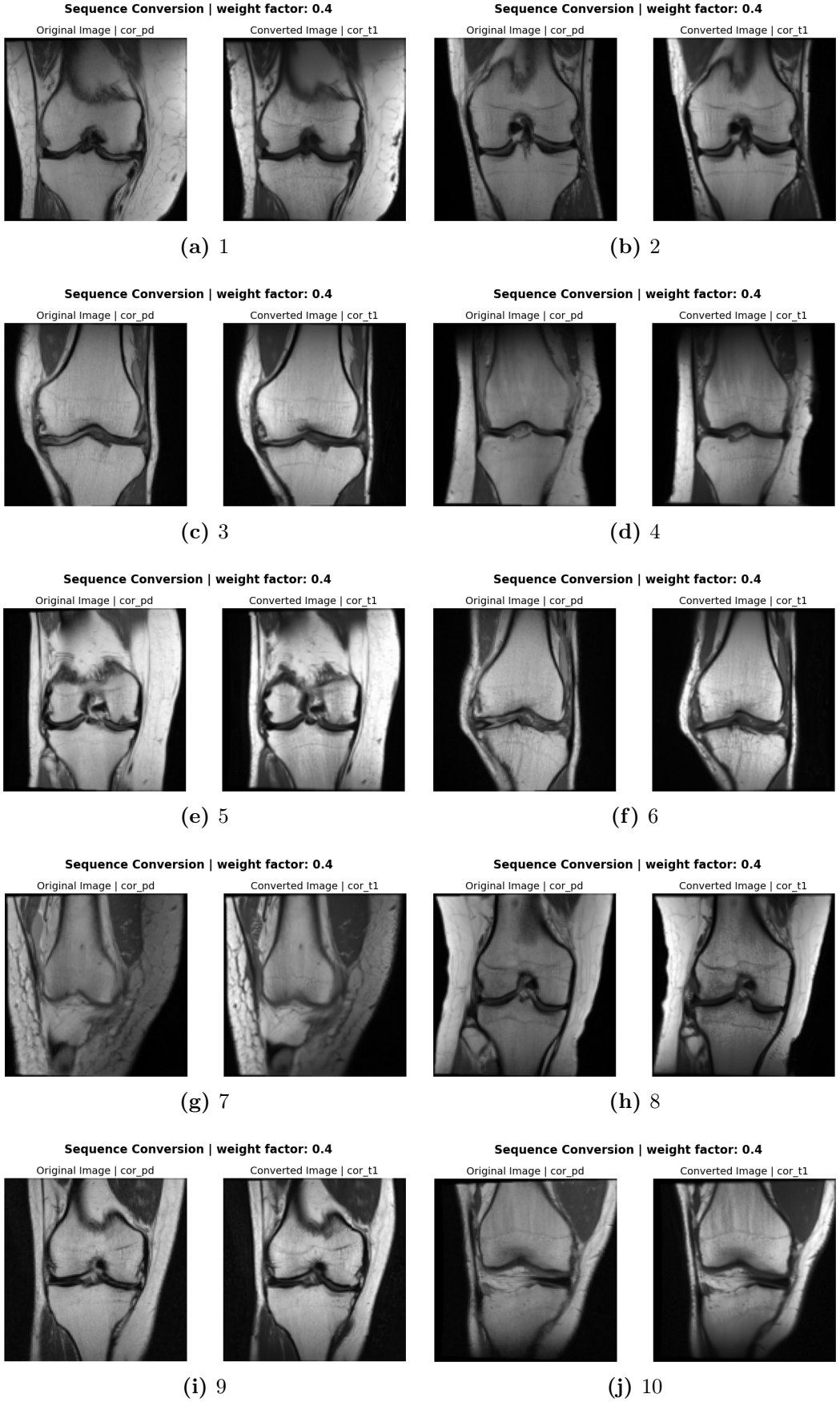
(c) 13 **(d)** 14

(e) 15 **(f)** 16

(g) 17 **(h)** 18

(i) 19 **(j)** 20

Figure A.3: Sequence Conversion COR PD to COR PD FS



(a) 1 **(b)** 2

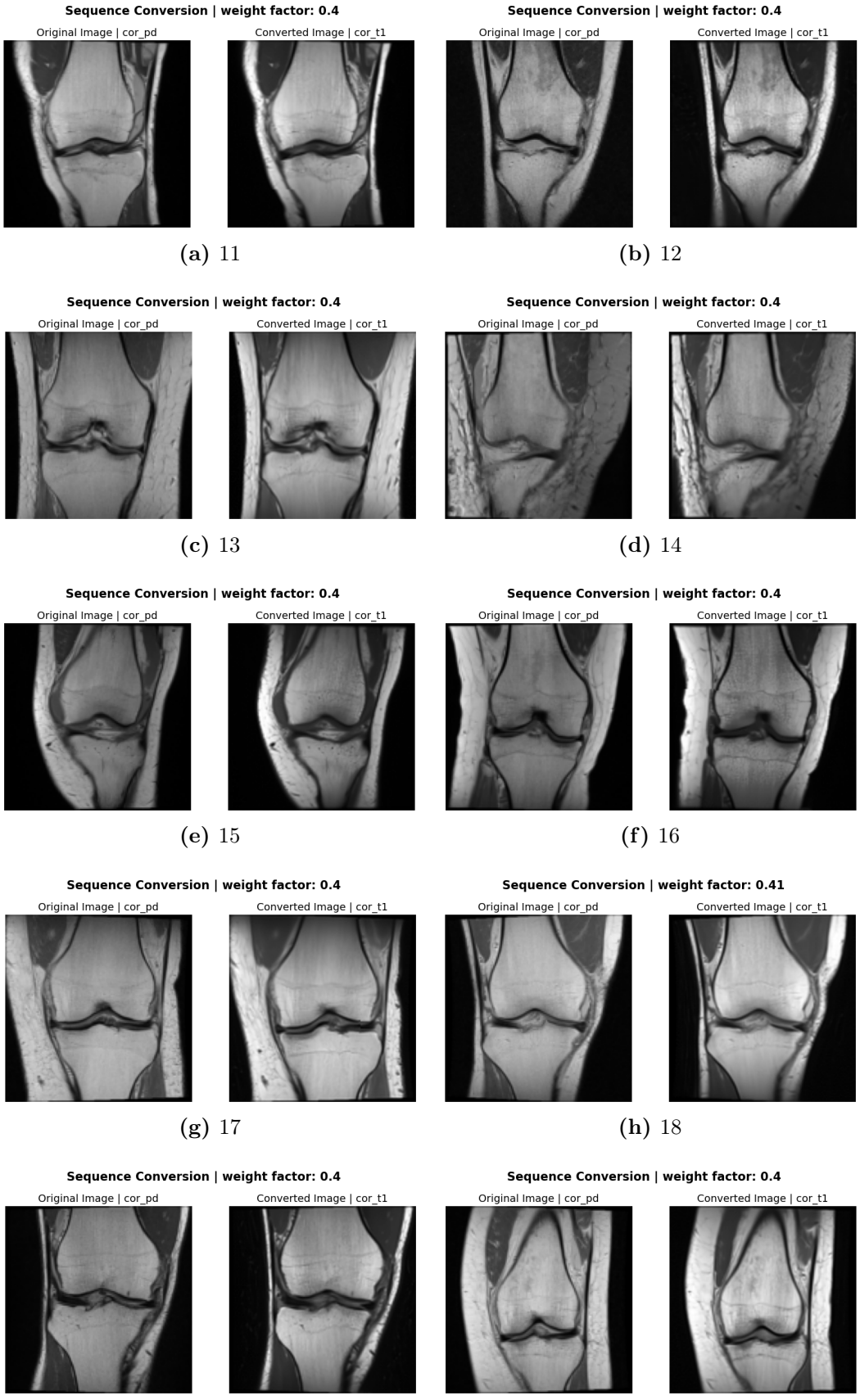
(c) 3 **(d)** 4

(e) 5 **(f)** 6

(g) 7 **(h)** 8

(i) 9 **(j)** 10

Figure A.4: Sequence Conversion COR PD to COR T1



(a) 11 **(b)** 12

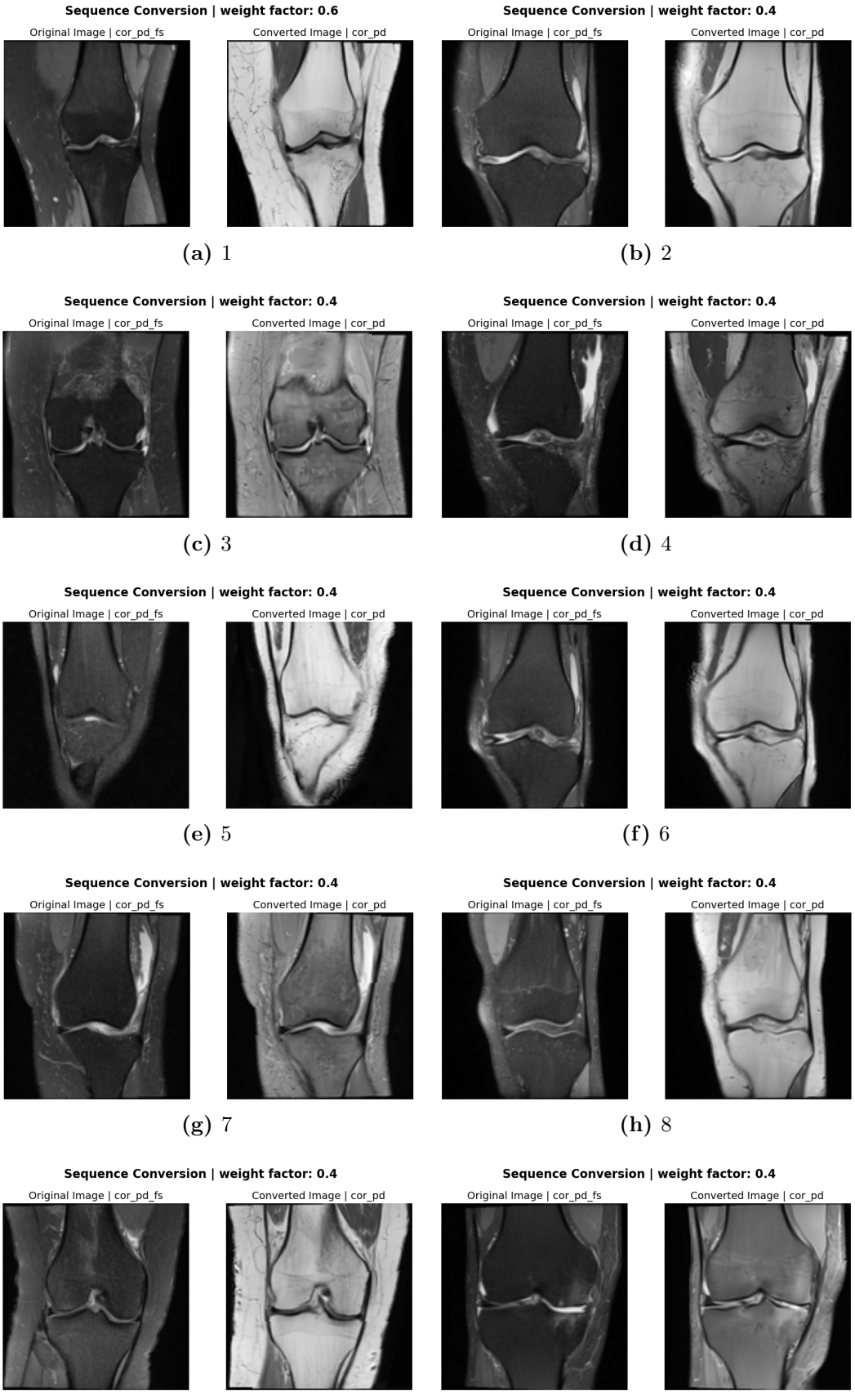
(c) 13 **(d)** 14

(e) 15 **(f)** 16

(g) 17 **(h)** 18

(i) 19 **(j)** 20

Figure A.5: Sequence Conversion COR PD to COR T1



(a) 1 **(b)** 2

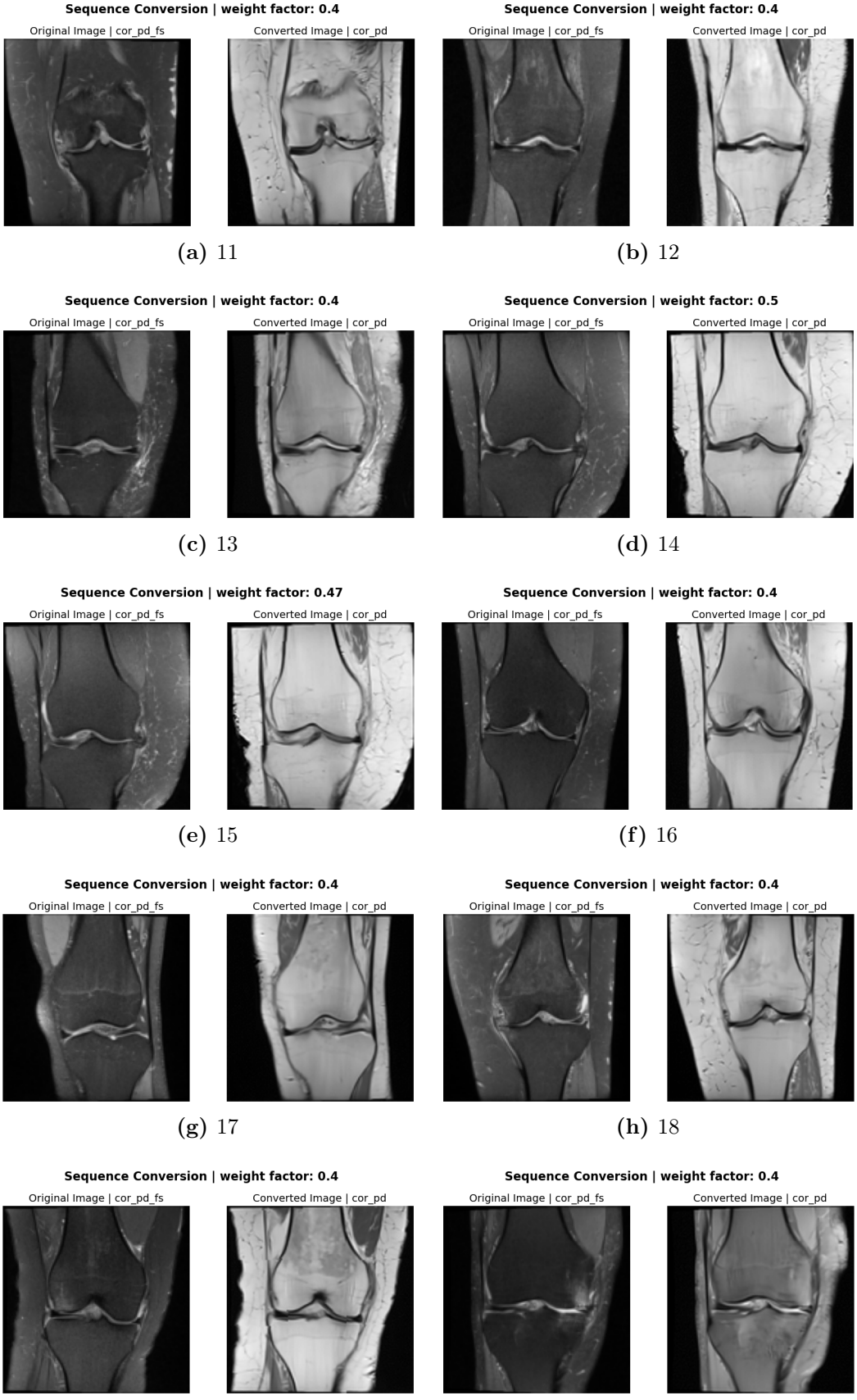
(c) 3 **(d)** 4

(e) 5 **(f)** 6

(g) 7 **(h)** 8

(i) 9 **(j)** 10

Figure A.6: Sequence Conversion COR PD FS to COR PD



(a) 11 **(b)** 12

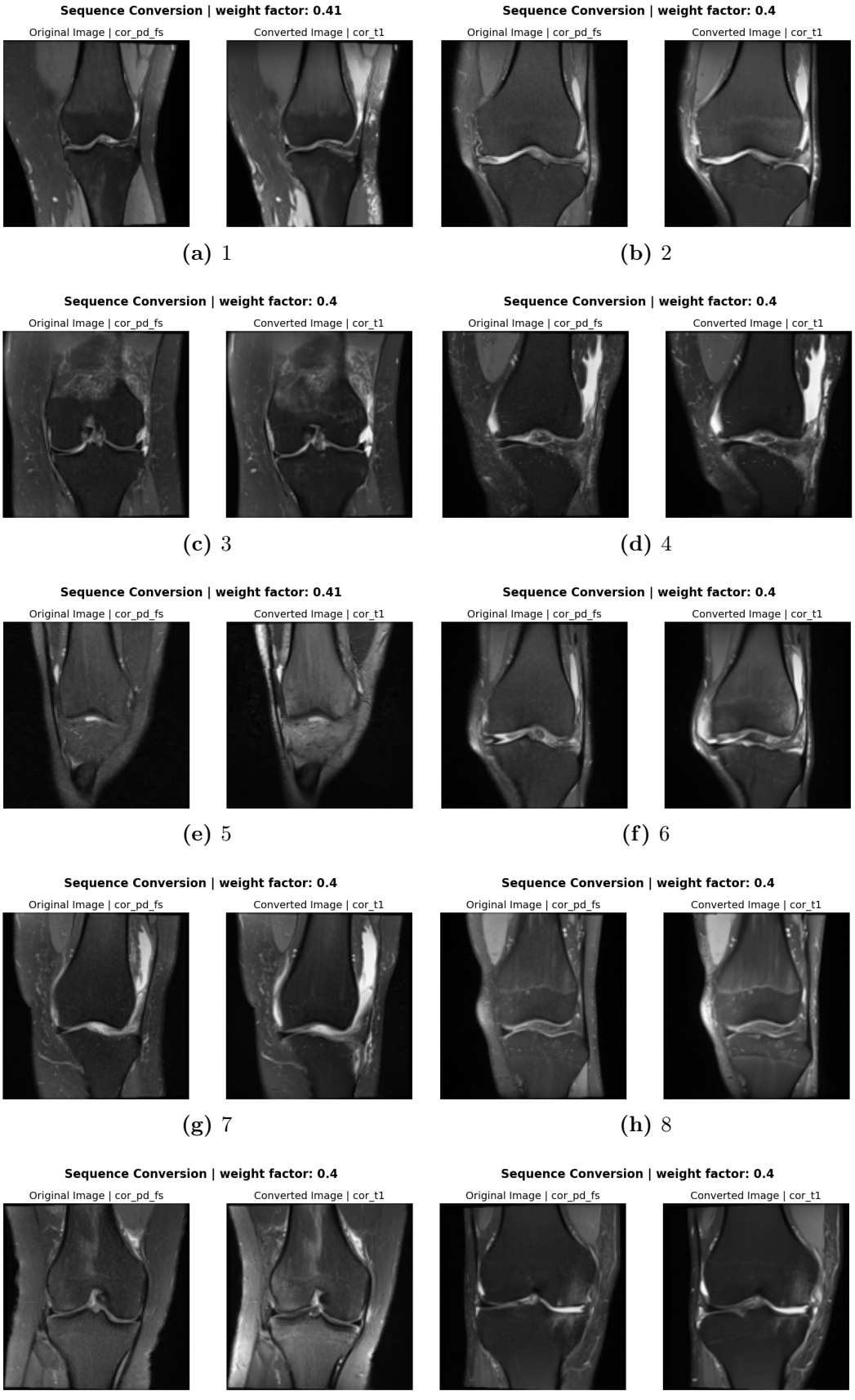
(c) 13 **(d)** 14

(e) 15 **(f)** 16

(g) 17 **(h)** 18

(i) 19 **(j)** 20

Figure A.7: Sequence Conversion COR PD FS to COR PD



(a) 1 **(b)** 2

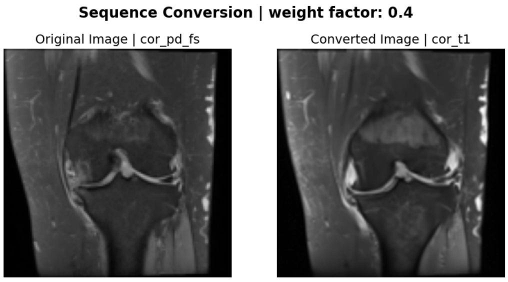
(c) 3 **(d)** 4

(e) 5 **(f)** 6

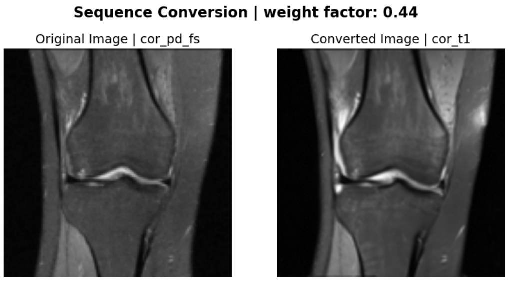
(g) 7 **(h)** 8

(i) 9 **(j)** 10

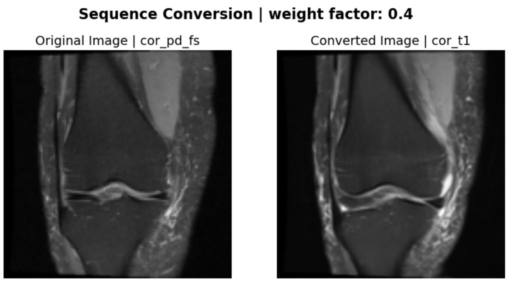
Figure A.8: Sequence Conversion COR PD FS to COR T1



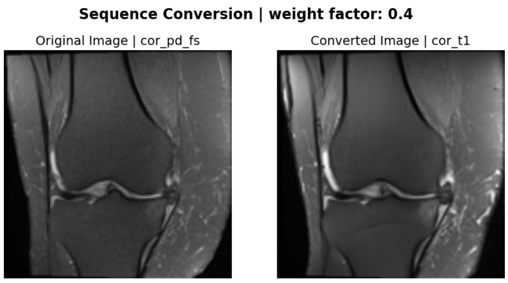
(a) 11



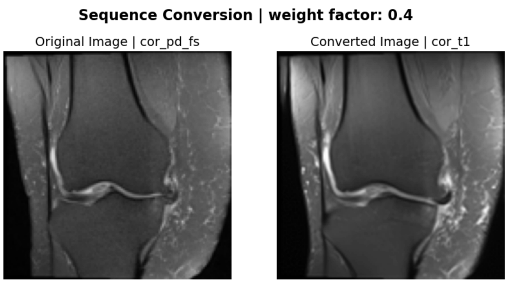
(b) 12



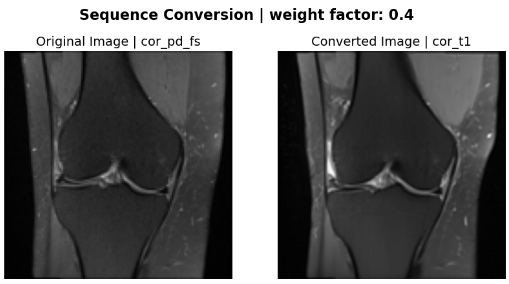
(c) 13



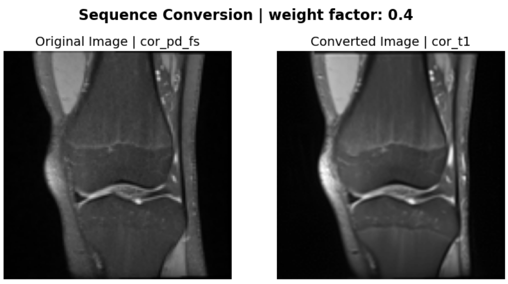
(d) 14



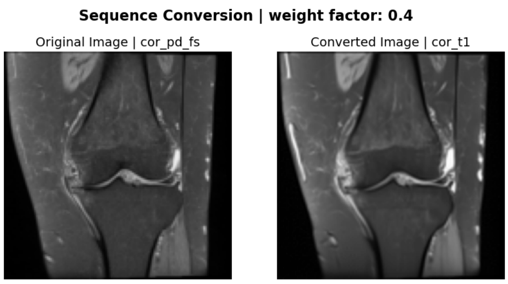
(e) 15



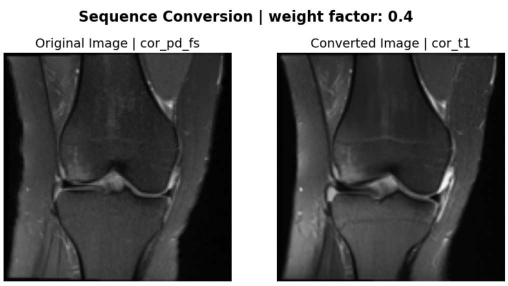
(f) 16



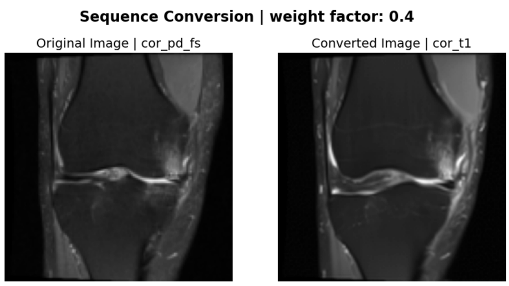
(g) 17



(h) 18

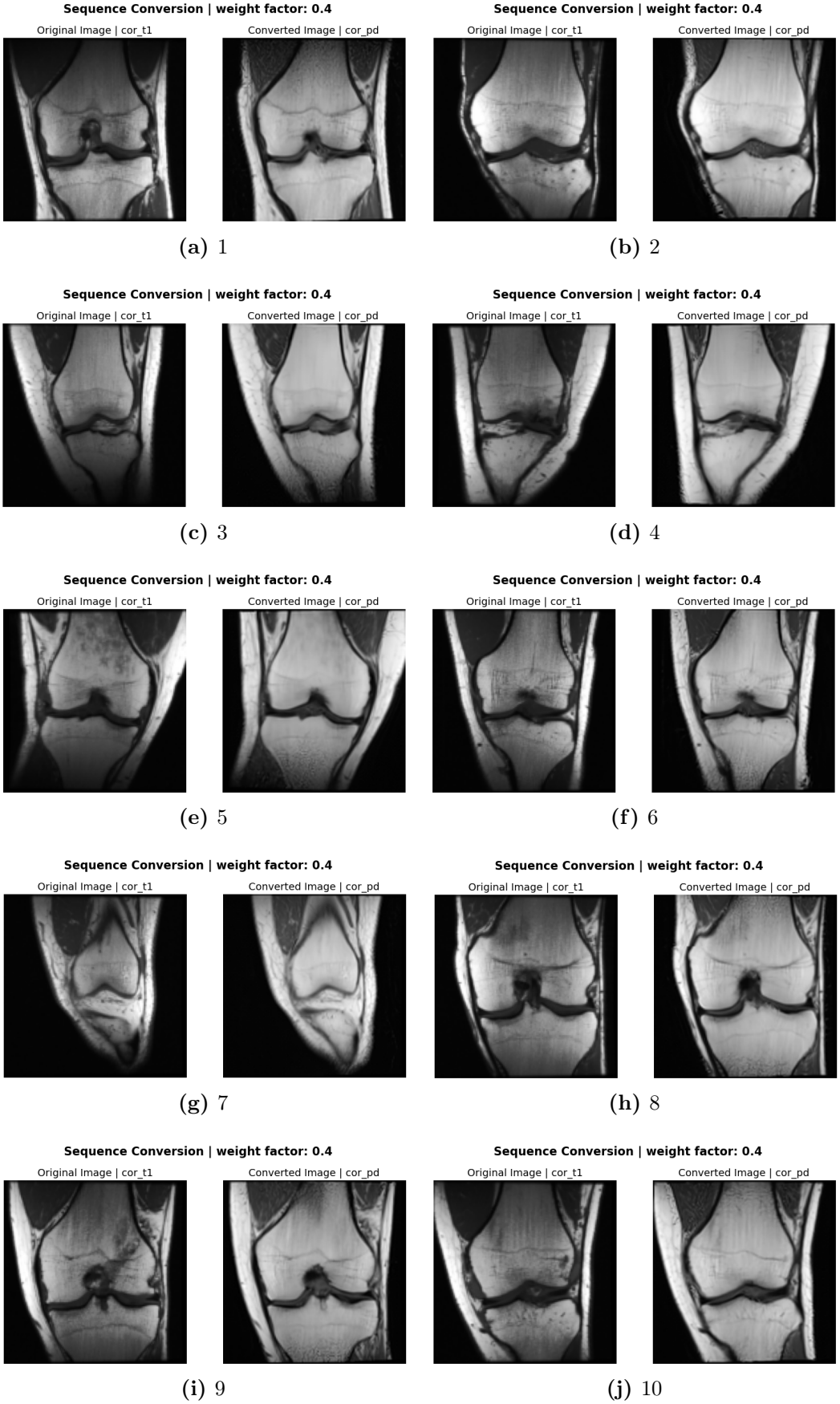


(i) 19



(j) 20

Figure A.9: Sequence Conversion COR PD FS to COR T1



(a) 1 **(b)** 2

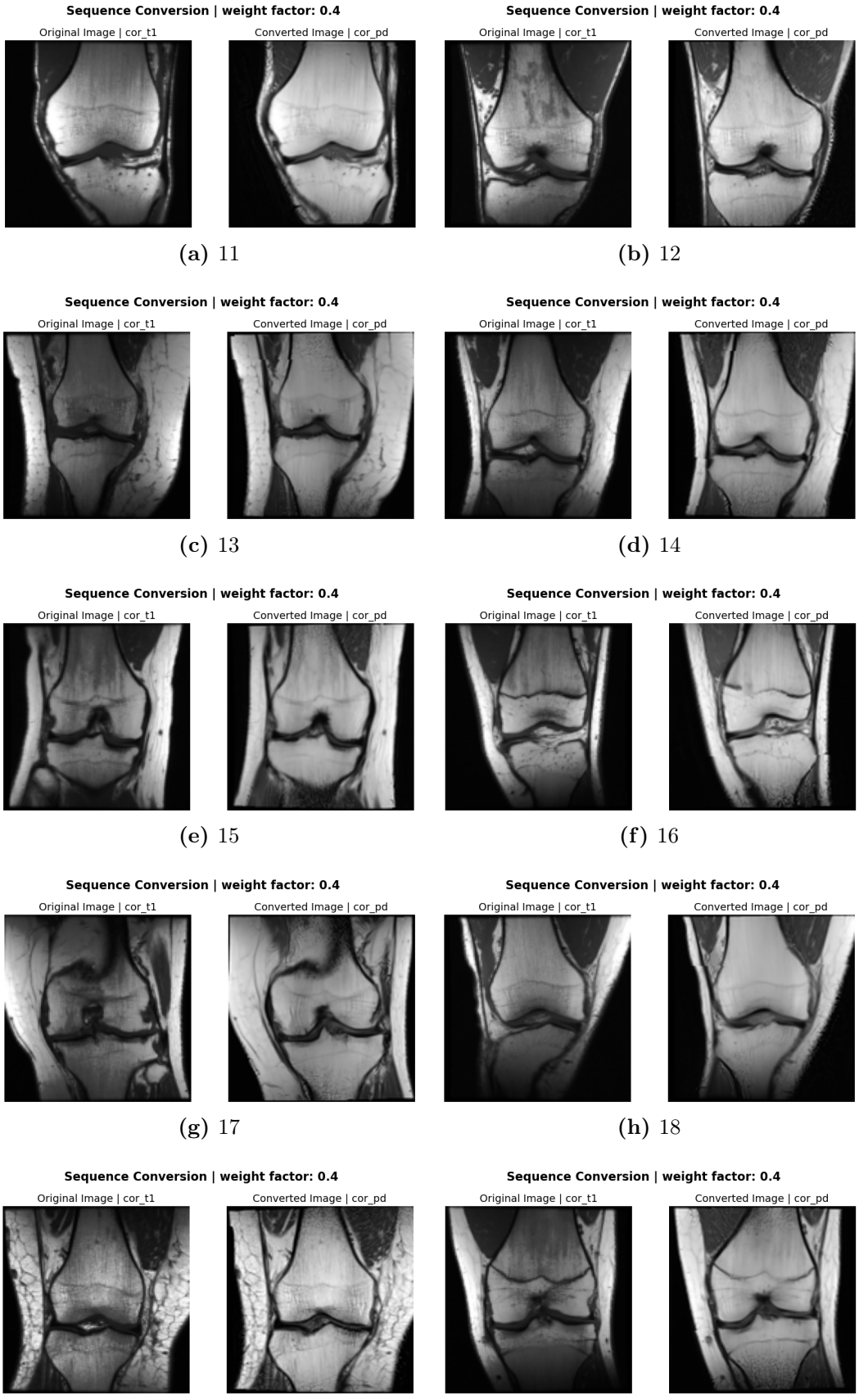
(c) 3 **(d)** 4

(e) 5 **(f)** 6

(g) 7 **(h)** 8

(i) 9 **(j)** 10

Figure A.10: Sequence Conversion COR T1 to COR PD



(a) 11 **(b)** 12

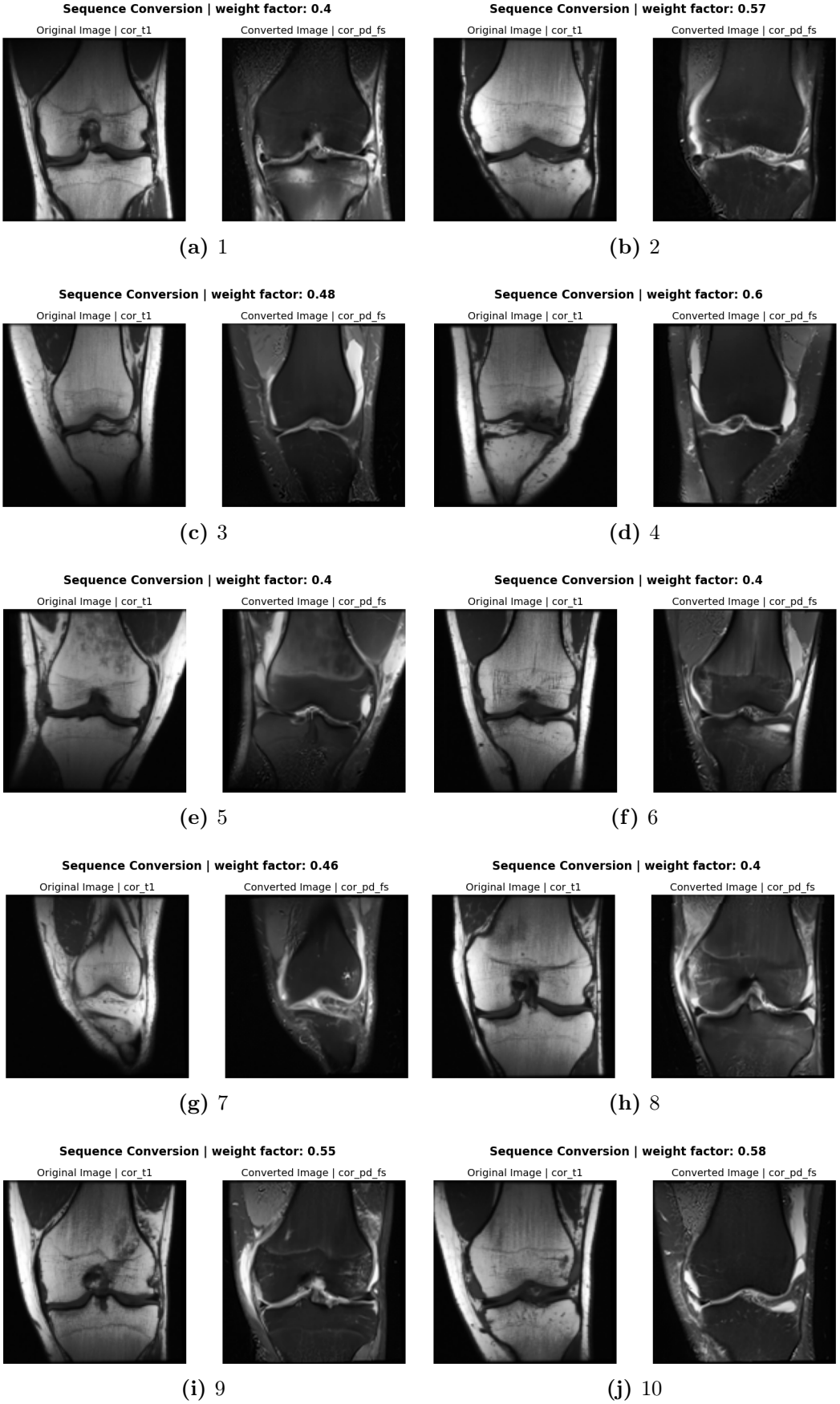
(c) 13 **(d)** 14

(e) 15 **(f)** 16

(g) 17 **(h)** 18

(i) 19 **(j)** 20

Figure A.11: Sequence Conversion COR T1 to COR PD



(a) 1 **(b)** 2

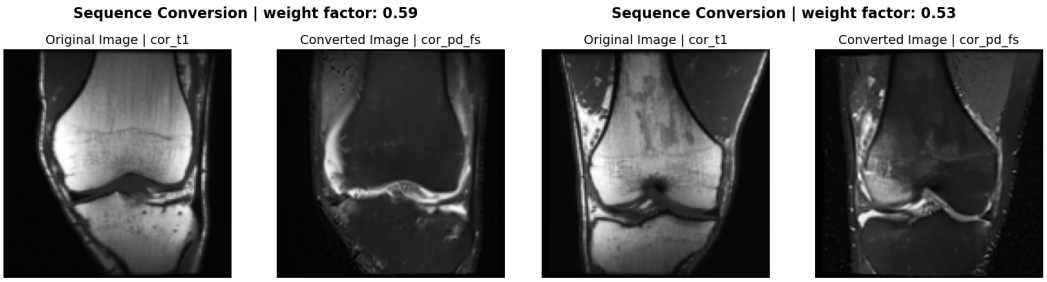
(c) 3 **(d)** 4

(e) 5 **(f)** 6

(g) 7 **(h)** 8

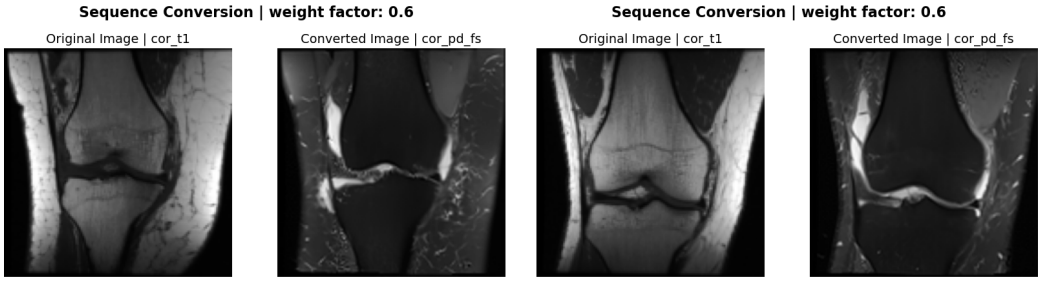
(i) 9 **(j)** 10

Figure A.12: Sequence Conversion COR T1 to COR PD FS



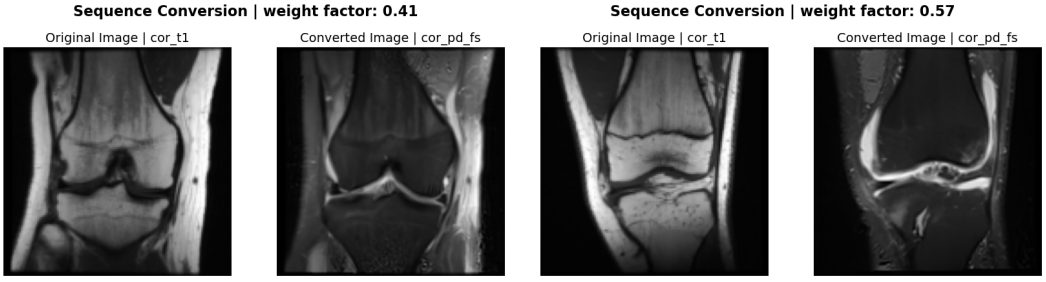
(a) 11

(b) 12



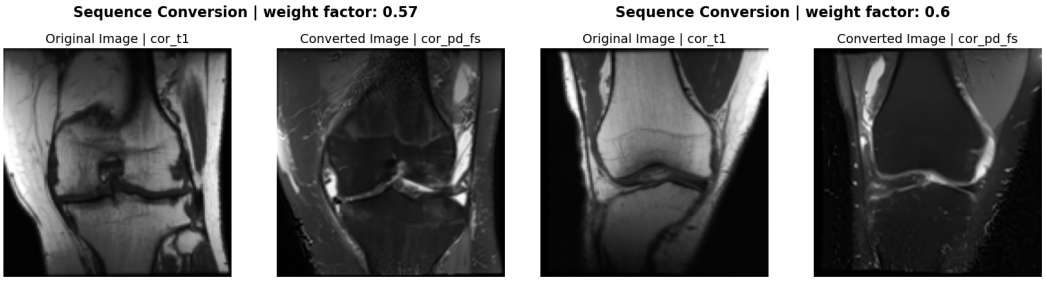
(c) 13

(d) 14



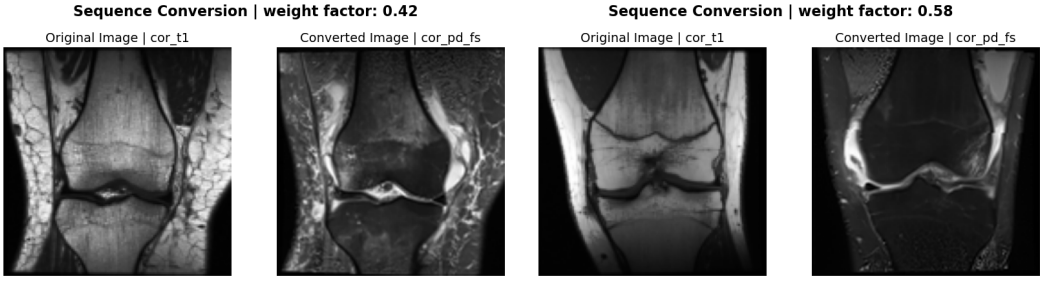
(e) 15

(f) 16



(g) 17

(h) 18



(i) 19

(j) 20

Figure A.13: Sequence Conversion COR T1 to COR PD FS