Imperial College London

MASTERS THESIS

Training unsupervised deep learning algorithms on 3D medical data

Author:
Laura HANU

Supervisor:
Dr. Anil A. BHARATH

Submitted in partial fulfillment of the requirements for the award of for the degree of Msc. Biomedical Engineering

in the

BICV Group
Department of Bioengineering, Imperial College London

Word count: 5293
September 15, 2017

Abstract

With the rise of deep learning applications on medical images, there is still unexplored potential of learning deep representations that could then be used to extract relevant features and assist and help healthcare specialists in diagnosis. This research project used three different unsupervised convolutional networks were explored on 2D and 3D MRI images of knees. A deep convolutional autoencoder was successfully trained on both 2D and 3D representations. Two generative adversarial networks (GAN), namely a deep convolutional GAN and the improved version of Wasserstein GAN (known as WGAN-GP), were then used to generate new samples of 2D MRI images. Whilst the DCGAN gave satisfactory results, but suffered from mode collapse, the WGAN-GP obtained very good results that concur with the original input data. Further work would include applying the deep representations learned on supervised tasks, such as classifying knee sizes.

Acknowledgements

I am deeply grateful to my supervisor Dr. Anil A. Bharath who provided invaluable guidance throughout the project. I would also like to thank Antonia Creswell for all the support, advice and great help with the code architectures and structures.

Contents

Chapter 1

Background

1.1 Introduction

Imaging techniques, such as CT and MRI, provide some of the most valuable clinical information in the medical field and are usually used for diagnosis or surgical planning. 2D medical images, and 3D medical images in particular, are more difficult to interpret and require expert opinion and take a vast amount of storage space, which makes manually categorizing and annotating medical images time-consuming and labour-intensive. Hence, there is a high demand for automating this process in order to better assist medical specialists [1] . As literature shows, recent developments in computer vision could offer a solution to these problems [2], which could prove to be a major milestone for intelligent healthcare.

This chapter aims to give an theoretical background to the field of deep learning, summarize recent research and developments in the network architectures with a focus on unsupervised learning and their application in the medical image analysis field.

1.2 Computer vision and deep learning

The goal of computer vision is to achieve automatic visual understanding of the input images or videos by re-interpreting high-dimensional data as numerical or symbolic information [3]. The field has seen major improvements with the breakthrough of deep learning in the last few years, which uses neural networks with multiple hidden layers to encode increasingly higher levels of abstraction. Since more and more images are digitally available online (such as ImageNet[4]), there has been an increase in the performance and optimisation speed of these algorithms.

1.3 Learning algorithm

Supervised learning uses a dataset D, which is formed of labelled examples of features that come in pairs, such that

$$D = \{\mathbf{x}, y\}_{n=1}^{N} \tag{1.1}$$

where $\mathbf{x}$ are the example features and y represents the label associated with $\mathbf{x}$. The goal is to find the optimal parameters, $\mathbf{\Omega}$, that minimize the loss function $L(y, \hat{y})$, where $\hat{y}$ is the output to the function $\mathrm{f}(\mathbf{x}; \mathbf{\Omega})$ that represents the model [5], [6].

Unsupervised learning aims to infer a function from data that is not labelled, and is thus concerned more with finding hidden patterns and creating and learning a representation of data that computers can understand. The chosen loss functions

are varied and are method-dependent[6], [7]. A caveat of this method, however, is that it is more difficult to determine the accuracy of the output of the model since there is no ground truth provided by labels.

Semi-supervised learning combines the 2 types of learning and generally uses a large amount of unlabeled data with a small amount of labeled data in order to perform a supervised task[8].

1.4 Neural Networks

Neural networks are composed of neural units or neurons, which have an activation function a and a set of parameters given by $\Omega = \{\mathbf{w}, b\}$, where $\mathbf{w}$ represents the weights and b represents the biases. The activation function a is composed of a linear combination of the input $\mathbf{x}$ and the parameters multiplied with a non-linear function σ, also knows as the transfer function:

$$a = \sigma(\mathbf{w}^T\mathbf{x} + b) \tag{1.2}$$

The most common transfer functions used are the sigmoid and the hyperbolic tangent function, although rectified linear units, knows as RELU, have become increasingly popular[9]. Deep neural networks have multiple layers that repeat the transformation in equation 1.2 and learn increasingly abstract features. When training the networks, the network is presented with a new set of samples, referred to as a batch during each iteration. This is where the parameters are updated through a process called back-propagation [10]. Initially, without a priori knowledge, the weights are represented by random values, which then are adjusted to minimize the cost function between the desired output and the actual output of the network. This is generally done by stochastic gradient descent method which chooses parameters that correspond to the minimum across the error surface [11]. There has recently been a resurgence in neural networks largely because of advancement in core computing systems, such as modern Graphics Processing Units (GPUs), which can perform complex mathematical operations in parallel with lower computational cost[7].

1.4.1 Convolutional neural networks (CNN)

Since they are highly parallelizable, CNNs have greatly benefited from the introduction of GPUs. They consist of convolutional layers that correspond to downsampled projections and higher perceptual levels. During each layer, a set of k kernels $\mathbf{W} = \{\mathbf{W_1}, \mathbf{W_2}, ..., \mathbf{W_k}\}$ is convolved with the input image, followed by adding the biases $B = \{b_1, b_2, ..., b_k\}$. This generates a feature map, $\mathbf{X_k}$, also knows as the receptive field. Then, a non-linear transformation is applied as follows for every convolutional layer c:

$$\mathbf{X}_k^c = \sigma(\mathbf{W}^T * \mathbf{X}^{c-1} + b) \tag{1.3}$$

The main advantage CNNs have over regular neural networks is that they drastically reduce the number of parameters at each layer, given they are adjusted only once for each instance of the kernel [5].

1.4.2 Autoencoders(AE)

One of the simplest unsupervised neural network is the autoencoder. Its aim is to reconstruct the input $\mathbf{x}$ on the output layer $\mathbf{x}'$ by typically using one hidden layer $\mathbf{h}$,

which encodes the input data into a lower-dimensional space, knows as the latent space:

$$\mathbf{h} = \sigma(\mathbf{W}_{\mathbf{x},h}\mathbf{x} + b_{x,h}) \tag{1.4}$$

The weight matrix $\mathbf{W}_{\mathbf{x},h}$, guides the input to the latent space alongside the bias matrix $b_{x,h}$, whereas $\mathbf{W}_{h,\mathbf{x}'}$ and $b_{h,x'}$ are used in decoding the hidden representation and computing the reconstruction [12], [13].

Convolutional Autoencoders (CAE)

An extension of the autoencoder, the convolutional autoencoder, uses convolutional layers to encode and decode the data, modelled after CNNs. This improves the performance of the normal autoencoder since convolutional layers use significantly less parameters. In addition, CAE help extract visual features from the images and increase the accuracy of the latent space representation. Consequently, the output can be used afterwards for a supervised task, such as training a classifier [1], [14]. However, It has been shown that a potential drawback is that some of the information is lost during the reconstruction process that results in increased blurriness of the output images [15].

1.4.3 Deep generative models

Deep generative models, an emerging subclass of unsupervised networks, have been proven to generate impressing results in a plethora of various applications, such as language and image generation [16]–[20], 3D shape generation [21]–[23] and semi-supervised learning [8], [24]–[26]. This section will focus on generative adversarial networks and will introduce 2 different approaches: the Deep Convolutional Generative Adversarial Networks (DCGAN) and the Wasserstein Generative Adversarial Network (WGAN).

Generative Adversarial Networks(GAN)

Introduced in 2014 [24], Generative Adversarial Networks use 2 pairs of networks, namely a generator and a discriminator, that compete in a non-zero sum game. A generator takes random noise as an input and tries to generate new examples of the input data without having access to the data. The discriminator is trying to distinguish between the generated samples and the real images based on the ground truth provided by the input images. They are both trained concomitantly: whilst the discriminator trains to differentiate better between synthetic and genuine data, the generator trains to confuse the discriminator. More explicitly, the latter may freeze when it becomes sufficiently accurate allowing the generator to improve by trying to lower the accuracy of the discriminator [24], [27]. The generator's objective is to minimize the log probability of the discriminator to correctly identify whether a generated image is real or fake. This is equivalent to minimizing the Jensen-Shannon divergence(JSD) between the distributions of the generator and the real data [28], [29]. Because the generator suffers from a vanishing gradient problem when the discriminator becomes too strong, a second objective has been implemented to maximize the log probability of the discriminator accurately recognizing a generated sample as genuine [9], [30]. These objectives can be mathematically represented as such:

$$\min_G \max_D \mathcal{L}(G, D) = \mathbb{E}_{\mathbf{x}\sim P_r}[\log D(\mathbf{x})] + \mathbb{E}_{\hat{\mathbf{x}}\sim P_g}[1 - \log D(\hat{\mathbf{x}})] \tag{1.5}$$

where P_r and P_g are the real and generated distributions, $D(\mathbf{x})$ is the discriminator function obtained from the real samples $\mathbf{x}$, $\hat{\mathbf{x}} = G(\mathbf{z})$ is the generator function using the latent space $\mathbf{z}$ to produce samples $\hat{\mathbf{x}} \sim P_g$ and $D(\hat{\mathbf{x}})$ is the discriminator function obtained from the generator[31].

Deep Convolutional Generative Adversarial Networks(DCGAN)

Deep convolutional generative adversarial networks (DCGAN) use convolutional layers and fully connected layers to learn deeper representations with less computational cost. One of the main advantages of DCGAN is that they can be successfully trained on smaller quantities of training data and reused as feature extractors for supervised tasks [25], [26]. However, a notable shortcoming is that training a GAN is considered to be more of an art than science in finding the optimal hyperparameters. They are notoriously known for being difficult to train with no universally accepted formula and have been shown to face problems with mode hoping, vanishing gradients and/or mode collapse [31]–[33].

Wasserstein GAN (WGAN) + Improved WGAN

Proposed only in 2017, the Wasserstein generative adversarial networks are already getting popular in the field of generative models offering promising results that seem to have a better performance overall than the DCGAN [31], [32], [34]–[36]. Most network structures use the mean square error, however, this has been shown to be associate with over-smoothed edges and detail loss [31]. The Wasserstein distance, also known as the Earth-Mover distance, was proposed instead of the Jensen-Shannon divergence to compute the difference between distributions [31].

In the standard GANs, the discriminator and generator are trained to solve the min-max problem (equation 1.6), however, it has been argued that this leads to an unstable generator[33], [30], [32].

WGAN proposes to use the Earth Mover distance to compare data distributions instead since it is continuous and differentiable almost everywhere, unlike the Jensen-Shannon divergence [31]. Implementing a WGAN within an existing GAN is rather trivial and only a few modifications are necessary such as the removal of the last sigmoid layer in the discriminator, the clipping of the weights of the discriminator after each update between $w = [-c, c]$, where c is a constant, and the removal of the log from the loss functions of the discriminator[32]. Thus, the objective equation becomes:

$$\min_G \max_D \mathcal{L}(G, D) = \mathbb{E}_{\mathbf{x}\sim P_r}[D(\mathbf{x})] + \mathbb{E}_{\hat{\mathbf{x}}\sim P_g}[D(\hat{\mathbf{x}})] \tag{1.6}$$

Shortly after, an improved version knows as WGAN-GP was proposed ([37]) using gradient penalty instead of the weight-clipping method in order to accelerate the convergence:

$$\mathcal{L}_{WGAN}(D) = \mathbb{E}[D(\hat{\mathbf{x}})] - \mathbb{E}[D(\mathbf{x})] + \Lambda\mathbb{E}[\text{-}] \tag{1.7}$$

where Λ a constant parameter and - is the gradient penalty:

$$(\|(\nabla_{\hat{\mathbf{x}}} D(\hat{\mathbf{x}}))\|_2 - 1)^2$$

$\hat{\mathbf{x}}$ is sampled uniformly along straight line pairs of generated and real samples. The loss function for the generator is expressed as:

$$\mathcal{L}_{WGAN}(G) = -\mathbb{E}[D(\hat{\mathbf{x}})] \tag{1.8}$$

It has been proved that these improvements result in better performance, more stability, quicker convergence and higher quality of the images [37].

1.5 Relevant research on medical images

Although deep learning has raised notable interest in the medical community, it has not been as widely applied on medical data as on natural images due to several issues. First, the fact that training these networks requires large sets of images proves to be challenging with medical images due to security and privacy concerns. Second, even in the rare cases of available medical databases, they are voluminous and complex and are often not annotated, which often requires specialist opinion to do manually.

Most applications of deep learning in medical image analysis revolve around object or lesion classification, image segmentation, registration, content-based image retrieval or image generation or enhancement [11]. Esteva et al. [38] and Gulshan et al. [39] recently achieved near human expert performance using fine-tuned versions of pre-trained architectures. Janowczyk et al. [40] used Stacked Autoencoders to normalize histopathology images and Benou et al. [41] performed denoising in DCE-MRI images. Exploiting the nature of MRI data to leverage essential information, 3D implementations of CNNs have also been successfully applied on medical data in [42], [43] to classify patients with Alzheimer's and in[44] to assess survival in patients that have advanced gliomas. Yang et al. used CNNs to identify landmarks on the distal femur surface using three sets of 2D MRI images on 3 different planes [45].

However, CNNs seem to be the golden standard in recent years with a large majority over other methods [6], [11]. This thesis aims to explore unsupervised models on 2D and 3D MRI images.

1.6 Aim and objectives

The aim of this research project was to extend the application of novel deep learning methods, such as the CAE or the GAN architectures aforementioned, to MRI images in order to achieve a deep feature representation of the real input data that could consequentially be used in supervised tasks. To achieve this, the workload was divided into three main objectives. Firstly, an adequate dataset of images had to be created from the raw MRI images provided to be used as input to the networks. Secondly, three different types of unsupervised deep networks were used to train the two-dimensional MRI images of knees to comparatively observe the perceptual differences in the results. The third objective was to exploit the intrinsic volumetric nature of MRI images and apply the networks directly to the 3D volumes in order to create a more accurate representation of the MRI data.

Chapter 2

Methods

2.1 Data format

This study used a dataset of 3D DICOM MRI images corresponding to 180 patients from the Osteoarthritis Initiative (OAI) website, which is dedicated to provide public access to clinical and medical images of osteoarthritis in knees[1]. Each 3D image was comprised of 160 slides of 384x384 2D images, which can be seen in Figure 2.1[2].

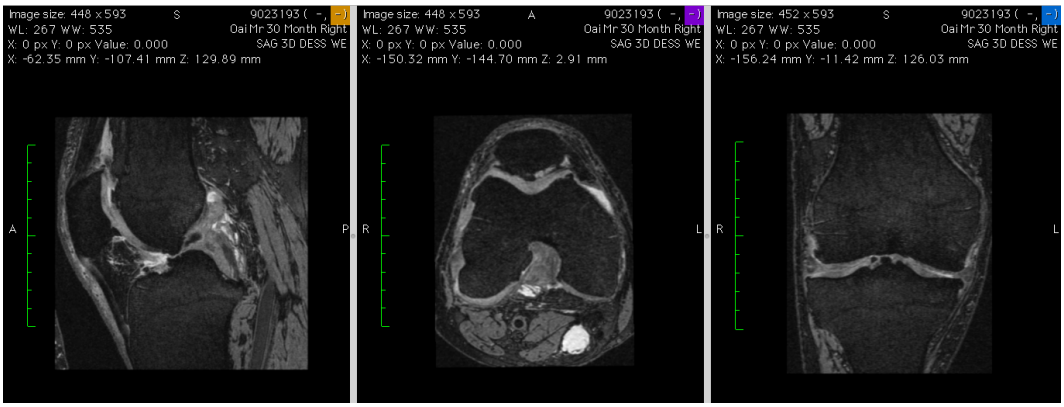

FIGURE 2.1: 3D view example of one patient in the sagittal, axial and coronal plane

2.2 Pre-processing

First, a Matlab function was created to convert the raw DICOM files to 3D matrices using the in-built Matlab DICOM reader function and resize the images to 64x64x160 pixels as seen in Figure 2.2.

Secondly, the intensity values were normalized to be between 0 and 1. Then, a python script was used to convert the Matlab files to Numpy arrays and create the final datasets. Conventionally, when training neural networks the following format is used for datasets:

$$\mathcal{D} = \{Nr.images, Nr.channels, Depth, Width, Height\} \tag{2.1}$$

Since grescale images were used in this case, the number of colour channels was 1. Two separate datasets were created: a 4D dataset which combined the depth and

[1] https://oai.epi-ucsf.org/

[2] 2D and 3D views obtained using OsiriX visualization software

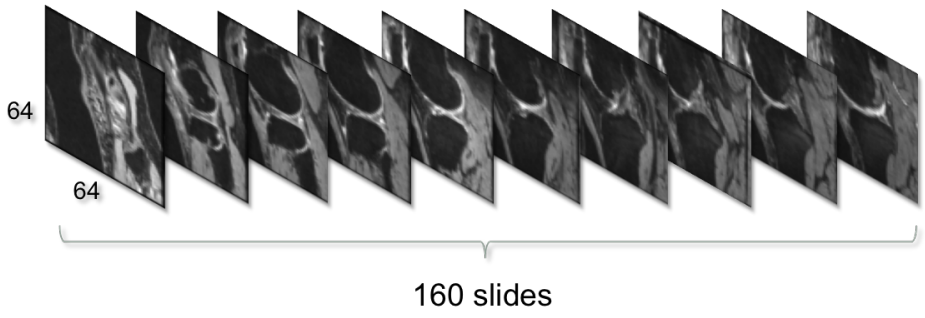

FIGURE 2.2: Depth view of the resized slides

number of patients into one dimension consisting of 28800 images, corresponding to:

$$\mathcal{D}_{\in\mathcal{D}} = \{28800, 1, 64, 64\} \tag{2.2}$$

and a 5D dataset that kept the depth of the images into a separate dimension:

$$\mathcal{D}_{\ni\mathcal{D}} = \{180, 1, 160, 64, 64\} \tag{2.3}$$

2.3 Backend Environment

A GPU operating on CUDA 8.0 was used for all computations. Theano and Lasagne were chosen as deep learning libraries since they offer a good combination of great compilation speed, automatic symbolic differentiation and a good model zoo[3]. The datasets were prepared using Python scripts in order to create the Numpy Arrays used as inputs for the neural networks used.

2.4 Explored Networks

3 different architectures were explored, namely the Convolutional Autoencoder(CAE), the Deep Convolutional Generative Adversarial Network(DCGAN) and the improved Wasserstein Generative Adversarial Network (WGAN-GP).

2.4.1 CAE

Architecture

Firstly, a simple convolutional autoencoder was built to learn a deep representation of the 2D dataset. The design of the architecture can be seen in Figure 2.3. The choice of network parameters such as filter size, padding, stride size and crop size was advised by the literature and reflects the most popular combination used in recent CNNs [1], [7]. 100 was chosen as the feature size in the latent space as a sensible compromise between the number of parameters and the number of features the model can learn. After a few empirical explorations, it was decided that 4 provides a suitable depth for the convolutional layers in both the encoder and the decoder. When developing the encoder, 4 convolutional layers were implemented. The Leaky Rectified Linear Unit (Leaky ReLU) with a negative slope of 0.2 was used as an activation function for first three convolutional layers. The sigmoid activation

[3]https://github.com/Lasagne/Recipes/tree/master/modelzoo

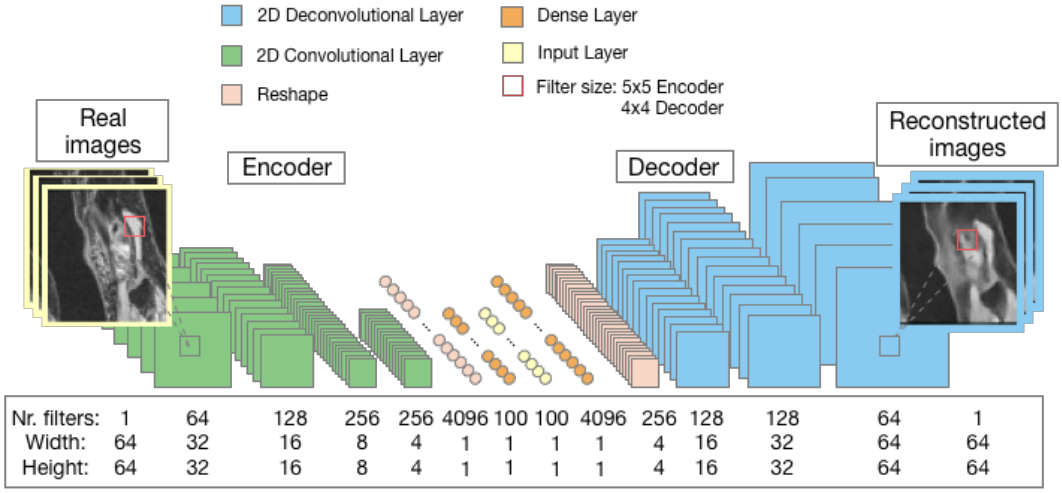

FIGURE 2.3: 2D CAE architecture

function was used in the last convolutional layer to add a non-linearity designed to increase the decoding difficulty. The chosen filter size was 5x5 with a stride of 2 and a pad of 2. For the last layer of the encoder, a dense layer was introduced as a fully connected layer with a number of units equal to the combined dimensions of the previous layer ($256x1x4x4 = 4096$). Full network details can be seen in Table 2.1 and Table 2.2.

TABLE 2.1: Architecture layour of the 2D encoder

Layer	Input	Filter	Stride	Pad	Activ.	Output
Input	-	-	-	-	-	1x1x64x64
Conv2D	1x1x64x64	5x5x64	2	2	Leaky ReLU	64x1x32x32
Conv2D	64x1x32x32	5x5x128	2	2	Leaky ReLU	128x1x16x16
Conv2D	128x1x16x16	5x5x256	2	2	Leaky ReLU	256x1x8x8
Conv2D	256x1x8x8	5x5x256	2	2	Leaky ReLU	256x1x4x4
Reshape	256x1x4x4	-	-	-	-	1x4096
Dense	1x4096	1x1x100	-	-	sigmoid	1x100

In the case of the decoder, a fully connected dense layer with 100 units (equal to the latent space) was added, followed by 4 deconvolutional layers using ReLU as activation function and the sigmoid activation function in the last layer.

TABLE 2.2: Architecture layout of the 2D decoder

Layer	Input	Filter	Stride	Crop	Activ.	Output
Input	-	-	-	-	-	1x100
Dense	1x100	1x1x4096	-	-	-	1x4096
Reshape	1x4096	-	-	-	-	256x1x4x4
Deconv2D	256x1x8x8	4x4x256	2	1	ReLU	128x1x16x16
Deconv2D	128x1x16x16	4x4x256	2	1	ReLU	128x1x32x32
Deconv2D	128x1x32x32	4x4x128	2	1	ReLU	64x1x64x64
Deconv2D	64x1x64x64	4x4x1	2	1	sigmoid	1x1x64x64

To update to 3D architecture, the 3D dataset was implemented with the volumetric information as displayed in Figure 2.4. The same layout was preserved as in the 2D case. 3D convolutional layers were used, however, Lasagne does not offer 3D deconvolutional layers yet, therefore, a new python function was created based on the development version of 3D Transposed Convolution source code on the Lasagne website [4]. A training set of 160 3D images was used, alongside a test set of the remaining 20 images.

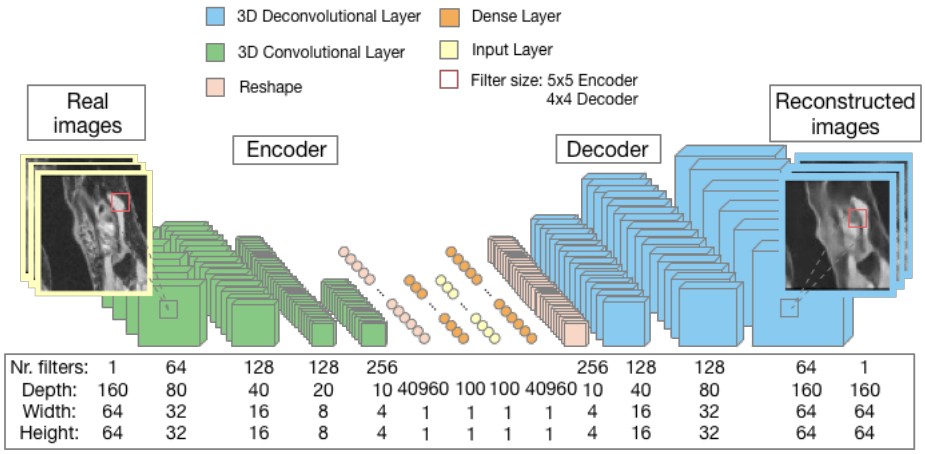

FIGURE 2.4: 3D CAE architecture

Main algorithm

Algorithm 1 CAE

1: **for** e in nr. of epochs **do**:
2: **for** n in nr. of batches **do**:
3: sample input data $\mathbf{x}$ and encode into:
3: $\mathbf{h} = \sigma(\mathbf{W}_{\mathbf{x},h}\mathbf{x} + b_{x,h})$:
3: reconstruct sample $\mathbf{x}' = \sigma(\mathbf{W}_{h,\mathbf{x}'}\mathbf{x} + b_{h,x'})$
6: minimizing loss using MSE:
7: $\mathcal{L}^{(n)} = \frac{1}{2n}\sum_{batch}(\mathbf{x}_n - \mathbf{x'}_n)^2$
8: update parameters $\Omega \leftarrow \mathcal{A}dam(\nabla_\Omega \mathcal{L}^{(n)}, \Omega, \alpha)$
12: **end for**
13: **end for**

When training the networks, the iterations were organised in epochs. During each epoch, the network is presented with a new set of samples, referred to as a batch. A larger batch side will provide a larger field of view at a directly proportional computational cost. The number of batches per epoch was calculated using: $Nr.\ batches = \frac{Nr.\ images}{batch\ size}$ The normalized 2D dataset was split into 2 distinct parts: the training set made of the first 27800 images and the test set consisting of the last 1000 images. The mean squared error (MSE) was used to minimize the loss between the encoding and decoding outputs as it can be seen in the pseudocode below. The

[4] http://lasagne.readthedocs.io/en/latest/modules/layers/conv.html

Adam optimizer was used to update the weights and the biases for both the encoding and the decoding layers.

2.4.2 DCGAN and WGAN-GP

Architecture

The GAN architecture used a reversed layout for the convolutional and deconvolutional layers employing upsampled deconvolutions first and downsampled convolutions second as seen in Figure 2.5. Both the DCGAN and the WGAN-GP net-

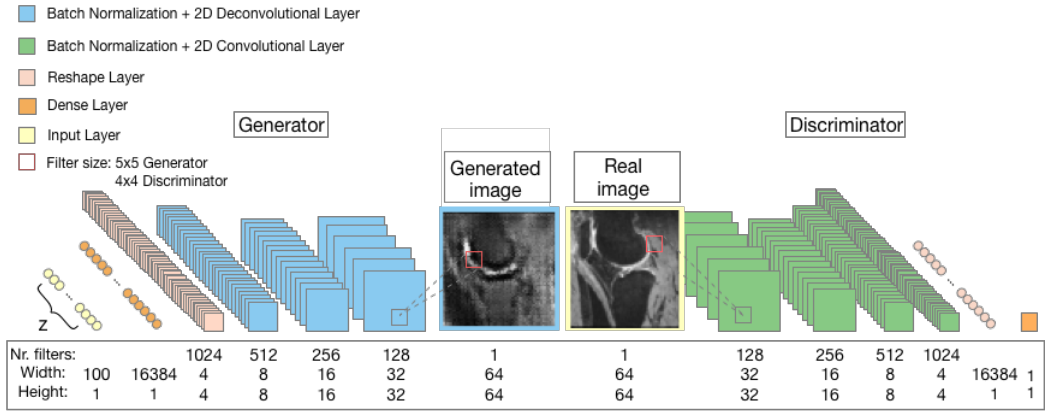

FIGURE 2.5: GAN architecture

works had the same layout presented in Tables 2.3 and 2.4 for the generator and the discriminator. The one exception was the last layer of the discriminator in the WGAN-GP, which had no activation function, as prescribed in literature.

TABLE 2.3: Architecture layout of the generator

Layer	Input	Filter	Stride	Crop	Activ.	Output
Input	-	-	-	-	-	1x100
Dense	1x100	1x1x16384	-	-	-	1x16384
Reshape	1x16384	-	-	-	-	1x1024x4x4
Deconv2D*	1x1024x4x4	5x5x512	2	1	ReLU	512x1x8x8
Deconv2D*	512x1x8x8	5x5x256	2	1	ReLU	256x1x16x16
Deconv2D*	512x1x16x16	5x5x128	2	1	ReLU	128x1x32x32
Deconv2D*	128x1x32x32	5x5x1	2	1	sigmoid	1x1x64x64

TABLE 2.4: Architecture layout of the discriminator

Layer	Input	Filter	Stride	Pad	Activ.	Output
Input	-	-	-	-	-	1x1x64x64
Conv2D*	1x1x64x64	5x5x128	2	2	L. ReLU	128x1x32x32
Conv2D*	128x1x32x32	5x5x256	2	2	L. ReLU	256x1x16x16
Conv2D*	256x1x16x16	5x5x512	2	2	L. ReLU	512x1x8x8
Conv2D*	512x1x8x8	5x5x1024	2	2	L. ReLU	1024x1x4x4
Reshape	1024x1x4x4	-	-	-	-	1x16384
Dense	1x16384	1x1x1	-	-	sigmoid*	1x100

Main algorithm

Although the architecture used were almost the same in both cases, the main differences between DCGAN and WGAN-GP can be seen in the way they compute their objective functions. As it can be seen in the algorithm below, a binary cross-entropy(bce) function was used on the discriminator in the case of the DCGAN when evaluating both the real and the generated dataset, which was then summed up to give the discriminator loss function. The Adam optimizer was used to update the discriminator parameters using the gradient to maximize the summed up loss function. The loss of the generator was given by the binary cross-entropy of the discriminator function and is then minimized using the Adam optimizer.

Algorithm 2 DCGAN

for e in nr of epochs **do**:
  for n in nr of batches **do**:
    sample real data $\mathbf{x} \sim P_r$ sample noise $\mathbf{z} \sim p(\mathbf{z})$
    $\hat{\mathbf{x}} \leftarrow G(\mathbf{z})$ $D(\mathbf{x}) \leftarrow (0,1)$
    update discriminator weights ascending the gradient:
    $\mathcal{L}^{(n)}(D) = bce\{D(\mathbf{x})\} + bce\{D(\hat{\mathbf{x}})\}$
    $w_D \leftarrow \mathcal{A}dam(\nabla_{\mathbf{d}} \frac{1}{n} \sum_{batch} \mathcal{L}^{(n)}(D), w, \alpha, \beta_1, \beta_2)$
    update generator weights down the gradient:
    $\mathcal{L}^{(n)}(G) = bce\{D(\hat{\mathbf{x}})\}$
    $w_G \leftarrow \mathcal{A}dam(\nabla_{\mathbf{g}} \frac{1}{n} \sum_{batch} \mathcal{L}^{(n)}(G), w, \alpha, \beta_1, \beta_2)$
  end for
end for

The most important distinctions between DCGAN and WGAN-GP is the introduction of the gradient penalty and the removal of the logarithm (used previously in the binary cross-entropy) when computing the discriminator function. The gradient penalty uses $\hat{\mathbf{x}}$, which takes straight lines across the real and generated samples in a uniform manner.

Algorithm 3 WGAN-GP

for e in nr of epochs **do**:
  for n in nr of batches **do**:
    sample real data $\mathbf{x} \sim P_r$ sample noise $\mathbf{z} \sim p(\mathbf{z})$ $\tilde{\mathbf{x}} \leftarrow G(\mathbf{z})$
    $\hat{\mathbf{x}} \leftarrow e\,\mathbf{x} + (1-e)\,\tilde{\mathbf{x}}$
    update discriminator weights:
    $\mathcal{L}^{(n)}(D) = D(\tilde{\mathbf{x}}) - D(\mathbf{x}) + \lambda\,(\|(\nabla_{\hat{\mathbf{x}}} D(\hat{\mathbf{x}}))\|_2 - 1)^2$
    $w_D \leftarrow \mathcal{A}dam(\nabla_{\mathbf{d}} \frac{1}{n} \sum_{batch} \mathcal{L}^{(n)}(D), w, \alpha, \beta_1, \beta_2)$
    update generator weights:
    $\mathcal{L}^{(n)}(G) = -D(\tilde{\mathbf{x}})$
    $w_G \leftarrow \mathcal{A}dam(\nabla_{\mathbf{g}} \frac{1}{n} \sum_{batch} \mathcal{L}^{(n)}(G), w, \alpha, \beta_1, \beta_2)$
  end for
end for

2.5 Hyper-parameter tuning

When it was not informed by the literature (as in the case of the WGAN-GP[31]), the choice of hyperparameters in terms of learning and decay rates was based on a empiric trial and error process. The values used for all the networks can be seen in Table 2.6. The batch size of 128 images gave the best results in the case of 2D images. However, when extended to 3D, the number of images is significantly smaller (180 compared to 28800) thus the batch size had to be modified accordingly. The overall number of parameters computed at each layer was also a factor in choosing the batch size given that a large number provides limitations on the GPU memory.

TABLE 2.5: Hyperparameters used

Network	**NZ**	**Batch size**	**Learning rate** α	**Decay rate** β_1	**Decay rate** β_2
CAE 2D	100	128	10^{-4}	0.5	-
CAE 3D	200	4	$5 * 10^{-5}$	0.5	-
DCGAN 2D	200	128	$5 * 10^{-5}$	0.5	0.9
WGAN 2D	200	128	$5 * 10^{-5}$	0.5	0.9

Chapter 3

Experimental Results

An example of 25 randomly selected images from the 2D dataset can be seen in the collage below as a representative for the ground truth.

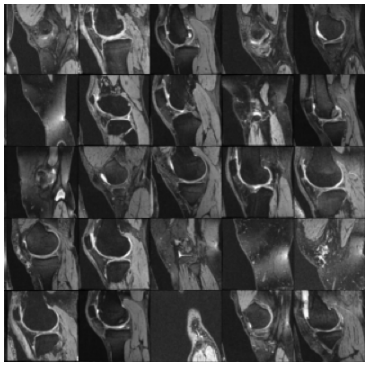

FIGURE 3.1: 25 randomly selected examples from the 2D dataset

3.1 CAE 2D

A sample of 6 images from the reconstruction results obtained by the two-dimensional convolutional autoencoder are displayed in Figure 3.2. The samples were taken every 20 slides from the first patient.

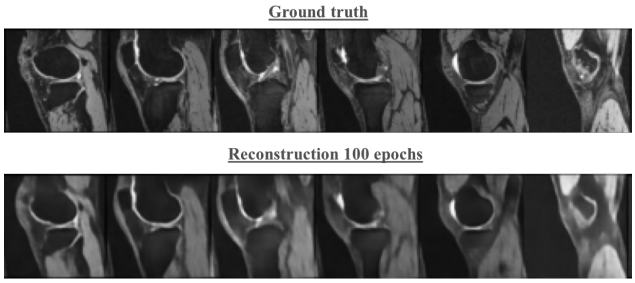

FIGURE 3.2: Reconstructed images after 100 epochs with a batchsize of 128 and $\alpha = 10^{-4}$

Figure 3.3. gives a plot of the loss functions for the training and test set. An example of the reconstructed image can be seen at the 20th, 40th and 80th epoch.

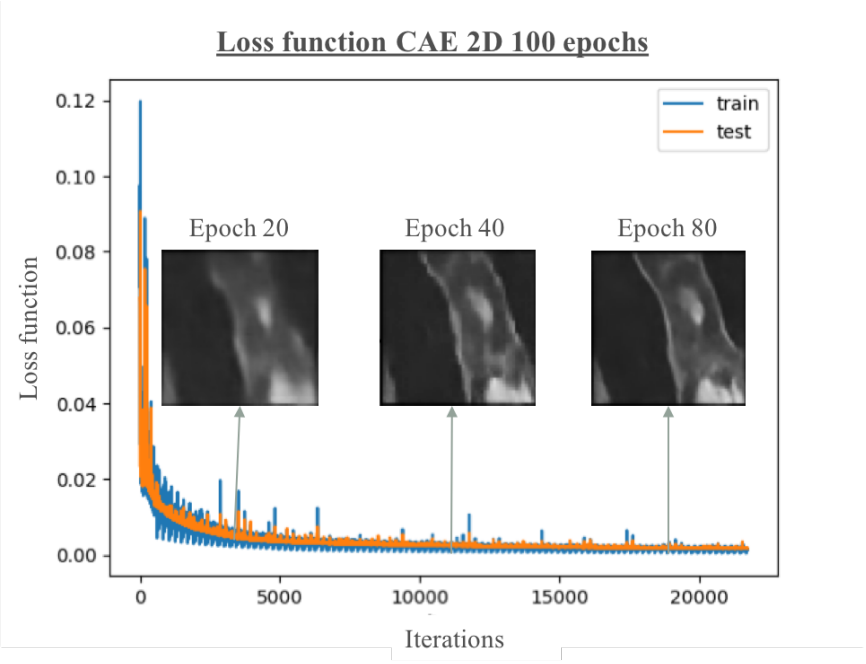

FIGURE 3.3: Loss function for the training and test set for 100 epochs

3.2 CAE 3D

To visualize the reconstructed 3D images, a selection of 3 different slide depths for 3 different patients at different training times were plotted to see the variation in the quality of the reconstruction. A comparison between the ground truth image and the reconstructed image can be seen in Figure 3.4.

The loss functions for the 3D reconstruction are plotted in the Figure 3.3.

3.3 DCGAN

A random selection of 9 generated images was chosen at 6 training times to demonstrate the evolution and convergence of the network (Figure 3.6), which can also be seen in the loss function plot (Figure 3.7).

3.4 WGAN-GP

The results obtained by the Wasserstein GAN with gradient penalty at different epochs can be observed in Figure 3.8. The network was trained on 35 epochs in batches of 128. Figure 3.9 offers an illustrative comparison between 100 real and generated images at the end of the iterations. The loss functions for 35 epochs can be seen in Figure 3.10.

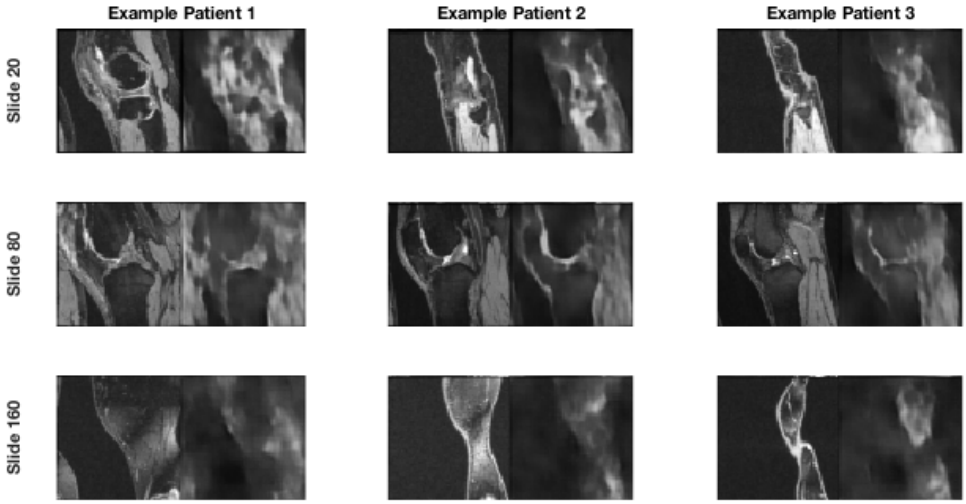

FIGURE 3.4: Ground truth (left image) and reconstruction examples (right image) from 3 random patients at 3 different slide depths (batch size=128, $\alpha = 10^{-4}$)

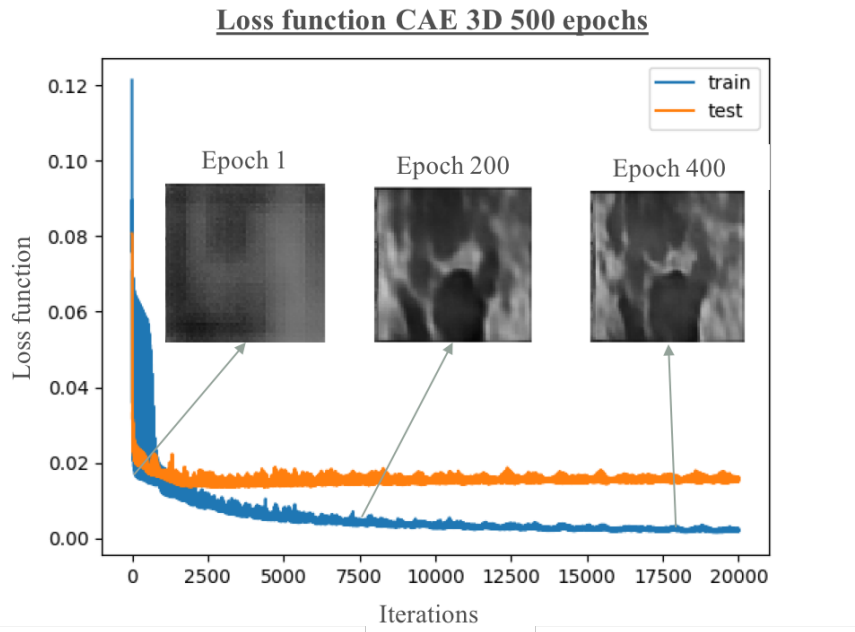

FIGURE 3.5: Loss functions for the test and training set for 500 epochs

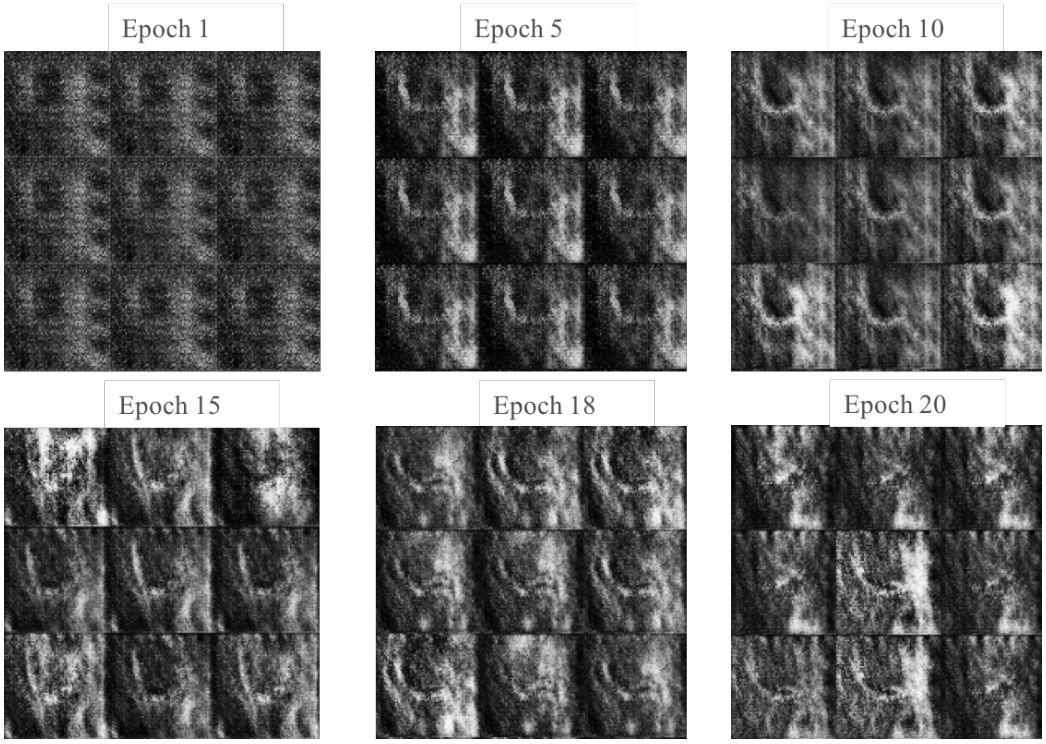

FIGURE 3.6: Generated examples for 20 epochs with batch size=128, $\alpha = 5 * 10^{-5}$, $\beta_1 = 0.5$, $\beta_2 = 0.9$

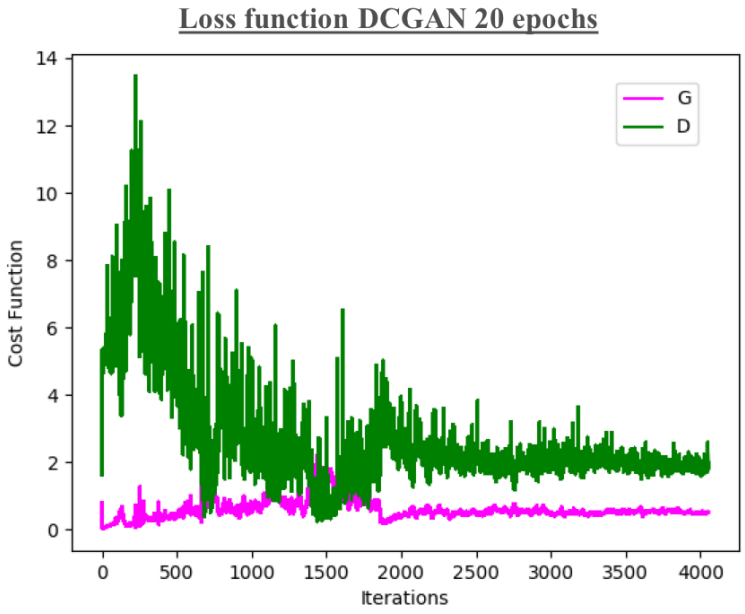

FIGURE 3.7: 3D view example of one patient in the sagittal, axial and coronal plane

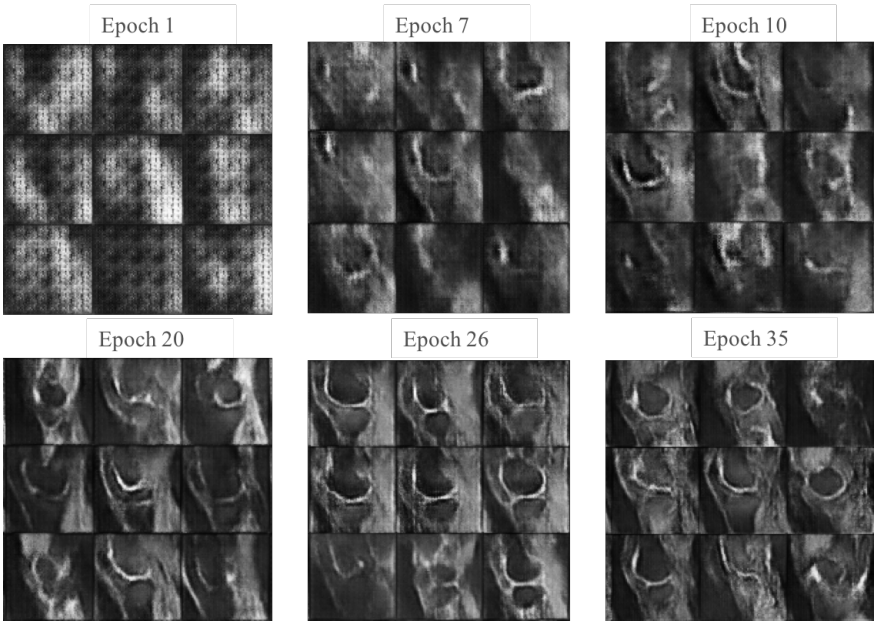

FIGURE 3.8: 3D view example of one patient in the sagittal, axial and coronal plane

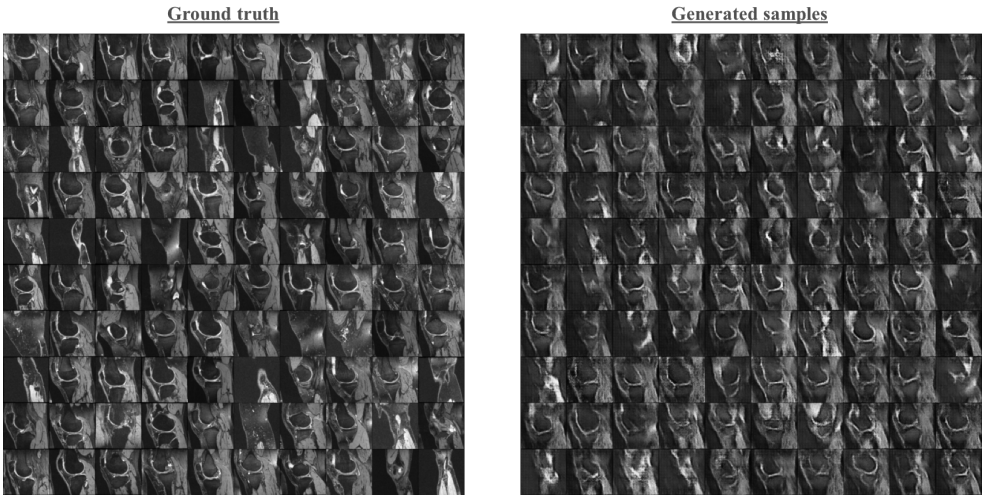

FIGURE 3.9: 3D view example of one patient in the sagittal, axial and coronal plane

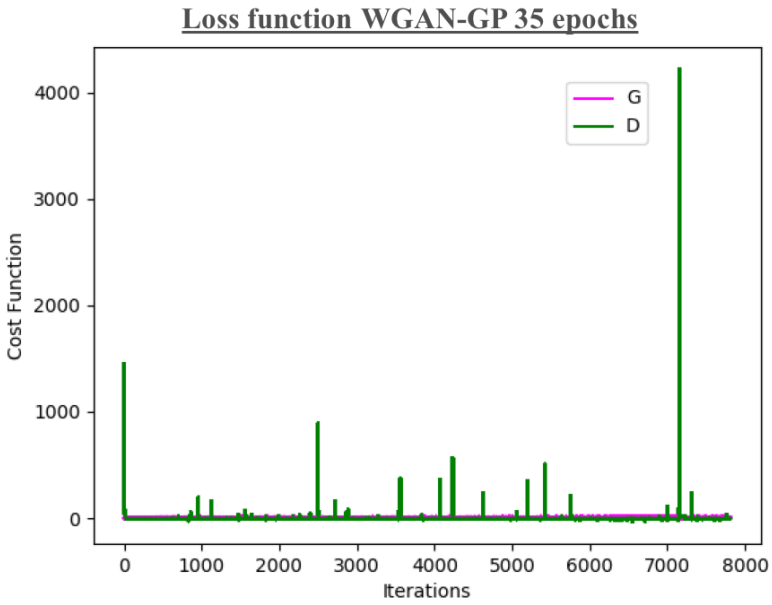

FIGURE 3.10: 3D view example of one patient in the sagittal, axial and coronal plane

Chapter 4

Discussion

Three different network types were explored in this research study: a convolutional autoencoder, a deep convolutional generative adversarial network and a Wasserstein generative adversarial network with gradient penalty. Given the volumetric nature of the data, 2 datasets (equations 2.2 and 2.3) were used in the case of the autoencoder to train on both 2D images, where the depth dimension was collapsed to enlarge the number of images, and the original 3D images. The first experiment used the 2D CAE to reconstruct the input using a batchsize of 128 and 22500 total iterations. After a few other smaller batch sizes were tried (32 and 64, which gave poorer results), the size of 128 was chosen as a good balance between the network field of view and compilation speed. The learning rate of $\alpha = 10^{-4}$ was used after the initial value of $\alpha = 10^{-2}$ led to an unfavourably quick convergence. As it be seen in Figure 3.2, the 2D reconstruction was successful with a negligible level of blurriness that may be due to insufficient training. After training for 100 epochs, it was decided that the network produced sufficiently satisfactory results to extent it to 3D. The second experiment used the extended 3D Convolutional Autoencoder architecture (Figure 2.4) to encode and decode 180 3D images. The network was trained on 22500 iterations and the same hyperparameters were used as in the 2D case with the exception of the batch size. Since the number of images was considerably smaller, a batch size of 4 was used to accommodate the high number of parameters (up to 10s of millions) at the deepest layers. It can be observed from Figure 3.4 and 3.5 that, despite the fact that the same number of iterations was used, the convergence is slower for 3D and the results are poorer. Furthermore, the loss functions for the training and test set diverge early on in the training meaning that the training set might start overfitting. As exptected, a closer look at the reconstructed images in Figure 3.4 shows that the performance of the decoder is less precise at the extremities of the volumes. The last two experiments were performed using the 2 GAN architectures on the 2D dataset: the DCGAN and the WGAN-GP. As it can be seen in Figure 3.6, DCGAN learns a decent representation of the real images using 4000 iterations, however, it suffers from mode collapse and an unstable convergence (Figure 3.7), especially on the discriminator. In comparison, WGAN-GP offers excellent results (Figure 3.8) and it converges quicker and in a more stable fashion (Figure 3.9). However, a few outliers in the discriminator cost function negatively influence the ability to accurately and reliably evaluate the convergence. Both networks used a batchsize of 128, $\alpha = 5 * 10^{-5}$, $\beta_1 = 0.5$, $\beta_2 = 0.9$. Out of all the experiments, it can be clearly seen that the WGAN-GP offers the best performance, learns the most features and can best reproduce the original data.

4.1 Limitations

One of the major limitation of training the aforementioned networks was the compromise between different architecture exploration, hyperparameter fine-tuning and longer training times, given that all have the potential to influence the quality of the results. Moreover, a common problem in unsupervised learning was experienced: the evaluation of the results accuracy was reduced to a subjective perceptual evaluation of the generated/reconstructed results. Another limitation was given by the GPU used, which offered memory constraints on the 3D architectures.

4.2 Future work

Further improvements could be obtained by using longer training times and a more extensive hyperparameter tuning. Additionally, a natural step forward would be to extend the DCGAN and WGAN-GP to a 3D architecture. If successful, the deepest generator output could be used for a supervised learning task, such as predicting knee size, given that a fixed mapping is provided.

4.3 Conclusion

The goal of this research project was to learn deep representations of MRI images by applying three different types of unsupervised deep neural networks. 2D MRI images of knees were successfully reconstructed using a 2D convolutional autoencoder. In order to achieve a better feature representation of the data, two types of generative adversarial networks were used to generate new examples of the input images. Whilst the DCGAN achieved a limited understanding of the real data with training and suffered from instability and mode collapse, the WGAN-GP succeeded in creating varied forged images that managed to encode different features of MRI depictions of knees. Subsequently, the convolutional autoencoder was upgraded to a three-dimensional architecture that satisfactorily encoded and decoded relevant features. Further training may be required to achieve better results. Given that out of all three types of network, the WGAN-GP offered the most promising results and learned the best representation, future work should focus on applying it to volumetric MRI data. The representation learned could then be used in supervised tasks, such as classification or prediction problems, that could potentially provide valuable assistance to medical experts.

Bibliography

[1] M. Chen, X. Shi, Y. Zhang, D. Wu, and M. Guizani, "Deep Features Learning for Medical Image Analysis with Convolutional Autoencoder Neural Network", *IEEE Transactions on Big Data*, vol. 7790, no. c, pp. 1–1, 2017, ISSN: 2332-7790. DOI: 10.1109/TBDATA.2017.2717439. [Online]. Available: http://ieeexplore.ieee.org/document/7954012/.

[2] R. Chauhan, H. Kaur, and R. Puri, "An Empirical Analysis of Unsupervised Learning Approach on Medical Databases", in *Emerging Trends in Electrical, Communications and Information Technologies: Proceedings of ICECIT-2015*, K. R. Attele, A. Kumar, V Sankar, N. V. Rao, and T. H. Sarma, Eds., Singapore: Springer Singapore, 2017, pp. 63–70. [Online]. Available: https://doi.org/10.1007/978-981-10-1540-3_7.

[3] D. Forsyth and J. Ponce, *Computer vision: A modern approach*. Upper Saddle River, NJ; London: Prentice Hall, 2011.

[4] A. Krizhevsky, I. Sutskever, and G. E. Hinton, "ImageNet Classification with Deep Convolutional Neural Networks", *Advances In Neural Information Processing Systems*, pp. 1–9, 2012. DOI: http://dx.doi.org/10.1016/j.protcy.2014.09.007. arXiv: 1102.0183.

[5] Y. LeCun, Y. Bengio, and G. Hinton, "Deep learning", *Nature*, vol. 521, no. 7553, pp. 436–444, May 2015, ISSN: 0028-0836. [Online]. Available: http://dx.doi.org/10.1038/nature14539http://10.0.4.14/nature14539.

[6] G. Litjens, T. Kooi, B. E. Bejnordi, A. A. A. Setio, F. Ciompi, M. Ghafoorian, J. A.W. M. van der Laak, B. van Ginneken, and C. I. Sánchez, "A Survey on Deep Learning in Medical Image Analysis", no. 1995, 2017. DOI: 10.1016/j.media.2017.07.005. [Online]. Available: http://arxiv.org/abs/1702.05747%0Ahttp://dx.doi.org/10.1016/j.media.2017.07.005.

[7] J. Schmidhuber, "Deep Learning in neural networks: An overview", *Neural Networks*, vol. 61, pp. 85–117, 2015, ISSN: 18792782. DOI: 10.1016/j.neunet.2014.09.003. [Online]. Available: http://dx.doi.org/10.1016/j.neunet.2014.09.003.

[8] D. P. Kingma, S. Mohamed, D. J. Rezende, and M. Welling, "Semi-supervised learning with deep generative models", in *Advances in Neural Information Processing Systems*, 2014, pp. 3581–3589.

[9] I. J. Goodfellow, D. Warde-Farley, M. Mirza, A. Courville, and Y. Bengio, "Maxout networks", *ArXiv preprint arXiv:1302.4389*, 2013.

[10] D. E. Rumelhart, G. E. Hinton, and R. J. Williams, "Learning representations by back-propagating errors", *Nature*, vol. 323, no. 6088, pp. 533–536, 1986. [Online]. Available: http://dx.doi.org/10.1038/323533a0.

[11] D. Ravi, C. Wong, F. Deligianni, M. Berthelot, J. Andreu-Perez, B. Lo, and G. Z. Yang, "Deep Learning for Health Informatics", *IEEE Journal of Biomedical and Health Informatics*, vol. 21, no. 1, pp. 4–21, 2017, ISSN: 21682194. DOI: 10.1109/JBHI.2016.2636665.

[12] L. Deng, M. Seltzer, D. Yu, A. Acero, A.-R. Mohamed, and G. Hinton, "Binary Coding of Speech Spectrograms Using a Deep Auto-encoder", no. September, pp. 1692–1695, 2010.

[13] G. E. Hinton and R. R. Salakhutdinov, "Reducing the dimensionality of data with neural networks", *Science*, vol. 313, no. 5786, pp. 504–507, 2006.

[14] B. Leng, S. Guo, X. Zhang, and Z. Xiong, "3D object retrieval with stacked local convolutional autoencoder", *Signal Processing*, vol. 112, pp. 119–128, 2015, ISSN: 01651684. DOI: 10.1016/j.sigpro.2014.09.005. [Online]. Available: http://dx.doi.org/10.1016/j.sigpro.2014.09.005.

[15] X.-J. Mao, C. Shen, and Y.-B. Yang, "Image restoration using convolutional auto-encoders with symmetric skip connections", *ArXiv preprint arXiv:1606.08921*, 2016.

[16] Z. Hu, Z. Yang, X. Liang, R. Salakhutdinov, and E. P. Xing, "Toward controlled generation of text", in *International Conference on Machine Learning*, 2017, pp. 1587–1596.

[17] A. Radford, L. Metz, and S. Chintala, "Unsupervised Representation Learning with Deep Convolutional Generative Adversarial Networks", pp. 1–16, 2015, ISSN: 0004-6361. DOI: 10.1051/0004-6361/201527329. [Online]. Available: http://arxiv.org/abs/1511.06434.

[18] X. Liang, Z. Hu, H. Zhang, C. Gan, and E. P. Xing, "Recurrent topic-transition gan for visual paragraph generation", *ArXiv preprint arXiv:1703.07022*, 2017.

[19] A. van den Oord, N. Kalchbrenner, L. Espeholt, O. Vinyals, A. Graves, *et al.*, "Conditional image generation with pixelcnn decoders", in *Advances in Neural Information Processing Systems*, 2016, pp. 4790–4798.

[20] A. Nguyen, J. Yosinski, Y. Bengio, A. Dosovitskiy, and J. Clune, "Plug & play generative networks: Conditional iterative generation of images in latent space", *ArXiv preprint arXiv:1612.00005*, 2016.

[21] A. A. Soltani, H. Huang, J. Wu, T. D. Kulkarni, and J. B. Tenenbaum, "Synthesizing 3d shapes via modeling multi-view depth maps and silhouettes with deep generative networks", CVPR, 2017.

[22] J. Wu, C. Zhang, T. Xue, B. Freeman, and J. Tenenbaum, "Learning a probabilistic latent space of object shapes via 3d generative-adversarial modeling", in *Advances in Neural Information Processing Systems*, 2016, pp. 82–90.

[23] J. Liu, F. Yu, and T. Funkhouser, "Interactive 3d modeling with a generative adversarial network", *ArXiv preprint arXiv:1706.05170*, 2017.

[24] T. Salimans, I. Goodfellow, W. Zaremba, V. Cheung, A. Radford, and X. Chen, "Improved Techniques for Training GANs", no. Nips, pp. 1–9, 2016, ISSN: 09246495. DOI: arXiv:1504.01391. [Online]. Available: http://arxiv.org/abs/1606.03498.

[25] E. Denton, S. Gross, and R. Fergus, "Semi-supervised learning with context-conditional generative adversarial networks", *ArXiv preprint arXiv:1611.06430*, 2016.

[26] A. Odena, "Semi-supervised learning with generative adversarial networks", *ArXiv preprint arXiv:1606.01583*, 2016.

[27] A. P. N. Creswell, T. White, V. Dumoulin, K. Arulkumaran, B. Sengupta, A. A. Bharath, and M. Ieee, "Generative Adversarial Networks : An Overview", no. April, pp. 1–12, 2017.

[28] I. Goodfellow, "Nips 2016 tutorial: Generative adversarial networks", *ArXiv preprint arXiv:1701.00160*, 2016.

[29] S. Nowozin, B. Cseke, and R. Tomioka, "F-gan: Training generative neural samplers using variational divergence minimization", in *Advances in Neural Information Processing Systems*, 2016, pp. 271–279.

[30] M. Arjovsky and L. Bottou, "Towards Principled Methods for Training Generative Adversarial Networks", pp. 1–17, 2017. DOI: 10.2507/daaam.scibook.2010.27. arXiv: 1701.04862. [Online]. Available: http://arxiv.org/abs/1701.04862.

[31] M. Arjovsky, S. Chintala, and L. Bottou, "Wasserstein GAN", 2017. [Online]. Available: http://arxiv.org/abs/1701.07875.

[32] S. Xiao, M. Farajtabar, X. Ye, J. Yan, L. Song, and H. Zha, "Wasserstein Learning of Deep Generative Point Process Models", 2017. [Online]. Available: http://arxiv.org/abs/1705.08051.

[33] Q. Yang, P. Yan, Y. Zhang, H. Yu, Y. Shi, X. Mou, M. K. Kalra, and G. Wang, "Low Dose CT Image Denoising Using a Generative Adversarial Network with Wasserstein Distance and Perceptual Loss", pp. 1–9, 2017. [Online]. Available: http://arxiv.org/abs/1708.00961.

[34] Z. Chen and Y. Tong, "Face super-resolution through wasserstein gans", *ArXiv preprint arXiv:1705.02438*, 2017.

[35] S. Liu, O. Bousquet, and K. Chaudhuri, "Approximation and convergence properties of generative adversarial learning", *ArXiv preprint arXiv:1705.08991*, 2017.

[36] Y. Kim, K. Zhang, A. M. Rush, Y. LeCun, *et al.*, "Adversarially regularized autoencoders for generating discrete structures", *ArXiv preprint arXiv:1706.04223*, 2017.

[37] I. Gulrajani, F. Ahmed, M. Arjovsky, V. Dumoulin, and A. Courville, "Improved Training of Wasserstein GANs", pp. 1–19, 2017, ISSN: 00308870. DOI: 10.1016/j.aqpro.2013.07.003. [Online]. Available: http://arxiv.org/abs/1704.00028.

[38] A. Esteva, B. Kuprel, R. A. Novoa, J. Ko, S. M. Swetter, H. M. Blau, and S. Thrun, "Dermatologist-level classification of skin cancer with deep neural networks", *Nature*, vol. 542, no. 7639, pp. 115–118, 2017.

[39] V. Gulshan, L. Peng, M. Coram, M. C. Stumpe, D. Wu, A. Narayanaswamy, S. Venugopalan, K. Widner, T. Madams, J. Cuadros, *et al.*, "Development and validation of a deep learning algorithm for detection of diabetic retinopathy in retinal fundus photographs", *Jama*, vol. 316, no. 22, pp. 2402–2410, 2016.

[40] A. Janowczyk, A. Basavanhally, and A. Madabhushi, "Stain normalization using sparse autoencoders (stanosa): Application to digital pathology", *Computerized Medical Imaging and Graphics*, vol. 57, pp. 50–61, 2017.

[41] A. Benou, R. Veksler, A. Friedman, and T. R. Raviv, "De-noising of contrast-enhanced mri sequences by an ensemble of expert deep neural networks", in *International Workshop on Large-Scale Annotation of Biomedical Data and Expert Label Synthesis*, Springer, 2016, pp. 95–110.

[42] E. Hosseini-Asl, G. Gimel'farb, and A. El-Baz, "Alzheimer's disease diagnostics by a deeply supervised adaptable 3d convolutional network", *ArXiv preprint arXiv:1607.00556*, 2016.

[43] A. Payan and G. Montana, "Predicting alzheimer's disease: A neuroimaging study with 3d convolutional neural networks", *ArXiv preprint arXiv:1502.02506*, 2015.

[44] D. Nie, X. Cao, Y. Gao, L. Wang, and D. Shen, "Estimating ct image from mri data using 3d fully convolutional networks", in *International Workshop on Large-Scale Annotation of Biomedical Data and Expert Label Synthesis*, Springer, 2016, pp. 170–178.

[45] D. Yang, S. Zhang, Z. Yan, C. Tan, K. Li, and D. Metaxas, "Automated anatomical landmark detection ondistal femur surface using convolutional neural network", in *Biomedical Imaging (ISBI), 2015 IEEE 12th International Symposium on*, IEEE, 2015, pp. 17–21.

[th]