Assignment of bachelor's thesis

Title:	Comparison of Generative Adversarial Networks on medical imaging data
Student:	Bc. Mark Sobolev
Supervisor:	Ing. Jakub Žitný
Study program:	Informatics
Branch / specialization:	Knowledge Engineering
Department:	Department of Applied Mathematics
Validity:	until the end of summer semester 2022/2023

Instructions

Research current state-of-the-art techniques that are used for augmenting medical imaging datasets. Compare the performance of chosen GAN architectures on CT and X-Ray data provided by the supervisor. Discuss the pros and cons of synthetic data used for classification and segmentation tasks. Present and compare the metrics one can use to evaluate GAN performance. Publish your prototype code and make sure your results are reproducible.

Electronically approved by Ing. Karel Klouda, Ph.D. on 3 March 2021 in Prague.

FACULTY
OF INFORMATION
TECHNOLOGY
CTU IN PRAGUE

Bachelor's thesis

Comparison of Generative Adversarial Networks on medical imaging data

Bc. Mark Sobolev

Department of Applied Mathematics
Supervisor: Ing. Jakub Žitný

May 13, 2021

Acknowledgements

Many thanks to my supervisor Ing. Jakub Žitný for his advice, willingness to help and quick response to my questions.

Declaration

I hereby declare that the presented thesis is my own work and that I have cited all sources of information in accordance with the Guideline for adhering to ethical principles when elaborating an academic final thesis.

I acknowledge that my thesis is subject to the rights and obligations stipulated by the Act No. 121/2000 Coll., the Copyright Act, as amended, in particular that the Czech Technical University in Prague has the right to conclude a license agreement on the utilization of this thesis as a school work under the provisions of Article 60 (1) of the Act.

In Prague on May 13, 2021 . . . . . .

Czech Technical University in Prague
Faculty of Information Technology

Citation of this thesis

Sobolev, Mark. *Comparison of Generative Adversarial Networks on medical imaging data.* Bachelor's thesis. Czech Technical University in Prague, Faculty of Information Technology, 2021.

Abstrakt

Cílem této práce je pokusit se použít nedávno navržený model generativní adversarialní sítě pro klasifikaci a segmentaci lékařských obrazů a porovnat výsledky s moderními modely GAN, které se v současné době používají pro rozšiřování lékařských dat.

Klíčová slova generativní adversariální sítě, augmentace dat, DCGAN, Pix2Pix, CycleGAN, generování medicínských obrazů, konvoluční neuronové sítě

Abstract

The aim of this work is to try to apply the recently proposed generative adversarial network model for the classification and segmentation of medical images and to compare the results with modern GAN models currently used for medical data augmentation.

Keywords generative adversarial networks, data augmentation, DCGAN, Pix2Pix, CycleGAN, medical image generation, convolutional neural networks

Contents

List of Figures

List of Tables

Introduction

Medical diagnostics is largely based on visual data, which are obtained by X-ray, computed tomography, dermoscopy, microscopy and other methods. For example, radiographs and CT scans are crucial for evaluation of bony structures and internal organs, while dermoscopic images help detect contagious and life threatening skin diseases such as skin cancer.

When diagnosing, two tasks are usually solved: classification and segmentation. Classification includes, for example, determining the malignancy of a tumor or the type of infection. Segmentation is the selection of certain objects in the image. These can be lungs, heart, kidneys, and so on.

Accuracy plays an important role in classification and segmentation. The health and lives of people depend on the correctness of the result. After all, an incorrect diagnosis can lead to both the death of the patient from an undetected pathology, and inadequate treatment, including unnecessary surgery. High level of accuracy may be achieved only with high level of training of medical specialists, attentiveness and patience. Therefore, human-based approaches are time intensive and laborous. Further improvement of computerized classification and segmentation methods is necessary.

Among modern automated methods, approaches based on machine learning look promising, since, unlike classical computer algorithms, they are capable of self-improvement. However, accessible amount of medical imaging data is often insufficient to train machine learning models in order to reach high accuracy. Data augmentation solves the problem of small number of samples available for training of such models. Many machine learning experts recognized that data augmentation is the way to improve human and automated medical diagnostics [1, 2].

There is no comprehensive guide on how to choose the most suitable model for medical imaging data augmentation or how to decide whether synthetic training data may be used for a particular classification or segmentation task.

The aim of this work is to compare different methods of medical imaging data augmentation with a focus on GAN-based methods and define their

limitations for training classification and segmentation models.

Chapter 1 contains theoretical aspects of machine learning necessary for understanding of principles modern generative adversarial networks are built on. Chapter 2 focuses on generative adversarial networks used in this work and metrics for quality assessment of the generated images. Chapter 3 presents datasets chosen for this work, evaluates images generated by a recently developed generative adversarial network model designed for fast training on small datasets [3] and compares it with other generative networks currently used for medical images classification and segmentation. Chapter 4 explains experiments on augmentation of selected datasets to improve classification and segmentation accuracy and compares the results with other current models.

Thesis's Objective

The main goal of this work is to identify modern GAN architectures most suitable for medical imaging data augmentation. In order to accomplish the main goal following partial goals must be achieved:

- Research current state-of-the-art GAN architectures that are used for augmenting medical imaging datasets.
- Present and compare the metrics one can use to evaluate GAN performance for generating medical data.
- Compare the performance of chosen GAN architectures on CT and X-Ray data provided by the supervisor.
- Discuss the pros and cons of using synthetic data for classification and segmentation tasks.
- Publish prototype code and make sure results are reproducible.

Chapter 1

Machine learning background

This chapter provides the information one needs to understand the machine learning principles on which this work is based.

1.1 Machine learning

Machine learning allows systems to adapt to a new data without exact directives [4].

Machine learning includes supervised, unsupervised, semi-supervised and self-supervised learning.

1.1.1 Supervised learning

Supervised learning starts with known pairs of available data samples y and labels x used to classify this data. The goal is to learn an underlying mapping between data. Based on the learned mapping it is possible to predict the labels for new samples.

1.1.2 Unsupervised learning

Unsupervised learning is performed on data samples without corresponding labels. The goal is to divide the data into clusters

Chapter 2

Generative adversarial networks

GAN (generative adversarial network) technique was firstly proposed by [5] and was based on the joint training of two models: generator G and discriminator D. G tries to mimic the original data x, while D distinguishes between generated and real samples by adjusting its function $D(x)$ denoting probability of x being a real sample. In order to learn G employs random noise vector $z \sim p_z(z)$. Training for G aims to maximize the probability of D confusing generated sample $G(z)$ with real one. Thus G and D participate in a minimax two-player game, where G works towards minimizing the value function $V(G, D)$, while D seeks to maximize it:

$$\min_G \max_D V(D,G) = E_{x\sim p_{data}(x)}[\log D(x)] + E_{z\sim p_z(z)}[\log (1 - D(G(z))] \quad (2.1)$$

The ultimate goal of the training is to make G create images from distribution p_g that match the statistical distribution p_{data} of real samples, when discriminator cannot guess, which image is real and which image is generated and $D(x) = \frac{1}{2}$.

It was proved in the paper that in order to reach global minimum for $\max_D V(D,G)$ generator should reproduce the target distribution of real data, i.e. to satisfy condition $p_g = p_{data}$ or in other words to reach zero Jensen-Shannon divergence $JSD(p_{data}||p_g)$.

[5] also proposed an algorithm for GAN training which 2 fundamental steps were ascend of the discriminator gradient and then descent of the generator gradient. The algorithm was compatible with every common learning method relying on gradients [5].

Another important finding from this work was that GANs can theoretically not only restore distributions $p(x)$, but also conditional distrbutions $p(x|c)$ in case, condition c is included as input for both generator and discriminator. Thus, the authors laid the foundation for creating conditional GANs, allowing to use generative networks not only for creating images, but also for translating images from one domain into another.

2.1 Convolutional neural networks

First GANs used multilayer perceptrons as both generator and discriminator [5]. Models were able to generate low resolution images, e.g. handwritten digits from CIFAR-10 dataset with size of 32×32 pixels. The synthesized images were, however, refered as of poor quality [6].

By this time, another class of neural networks, convolutional neural networks (CNNs), was already widely used for image processing [7]. Using 2 operations: discrete convolution and pooling, CNNs allowed to bring image classification to a new level of accuracy [8]. The reason convolution networks were so successful in image processing was because discrete convolution provided a way to preserve spatial structure of visual data.

The algorithm of discrete convolution is depicted in the figure 2.1. The data that goes through the convolution process are called input feature maps [7]. It is represented by pale blue lattice. A kernel matrix (dark blue grid) moves along the input feature map. To calculate the result of the convolution, called output feature map, it is necessary to multiply in pairs the elements of the input feature map and the elements of the kernel overlapping them, and then add the resulting products. It is possible to obtain multiple output feature maps from one input feature map by repeating the process with various kernels (Figure 2.2).

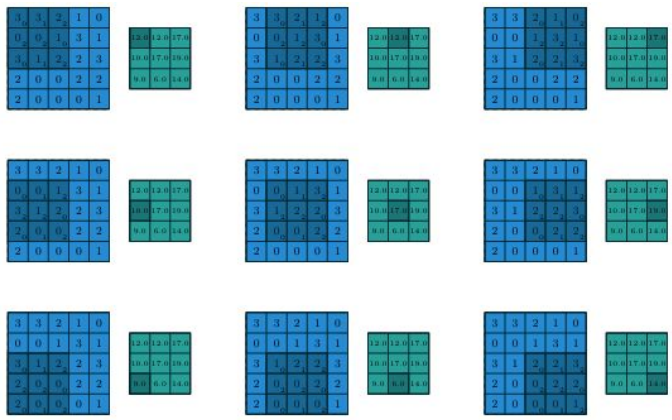

Figure 2.1: Calculating discrete convolution [7].

The shape of each convolutional kernel with n output feature maps, m input feature maps and size k_j of a dimension j may be described by tuple $(n, m, k_1, .., k_n)$. The output feature map size in dimension j depends on:

- input size i_j in the relevant dimension
- kernel size i_j in the relevant dimension

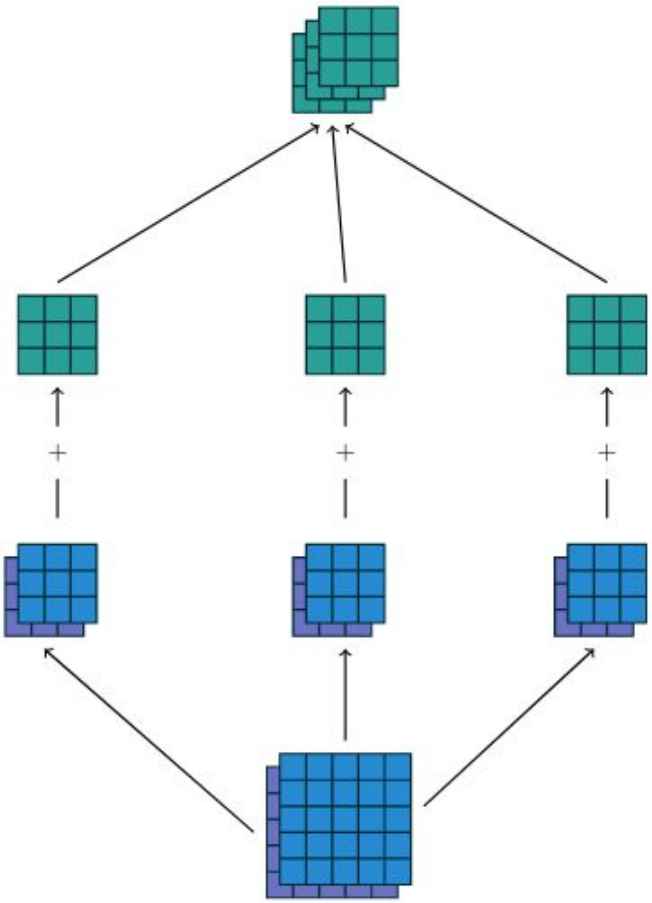

Figure 2.2: Obtaining 3 output feature maps from 2 input feature maps by applying a group w of 3 pairs of 3×3 kernels. The left branch represents applying kernels $w_{1,1}$ and $w_{1,2}$ to the first and second feature maps correspondingly and adding up the results, thereby forming the first output feature map [7].

- interval among successive locations of the kernel s_j, also called stride, when calculating the value of each element of the output feature map
- amount of elements (often zeros) p_j, also called padding, appended to the input feature map from both sides in the relevant dimension in order to adjust the output feature map size

Example of convolution with strides and padding may be seen in Figure 2.3.

The pooling operation reduces the size of the input feature map using a transformation that in a certain way characterizes by one value the area to which it is applied. This can be the calculation of the mean or the highest

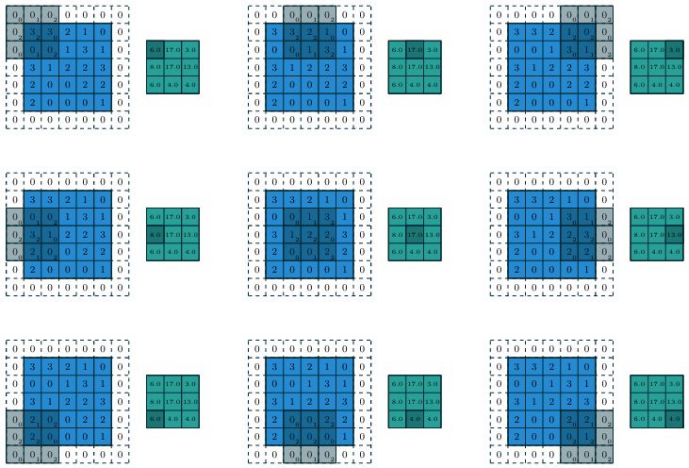

Figure 2.3: Convolving an input feature map with $i_1 = i_2 = 5$ and kernel of size $k_1 = k_2 = 3$ with zero padding $p_1 = p_2 = 1$. Stride $s_1 = s_2 = 2$ means that the convolutional kernel moves 2 steps instead of one between 2 consecutive convolutions [7].

value. Like convolution, it is sequentially applied to all regions of the input feature map, as shown in the figure 2.4.

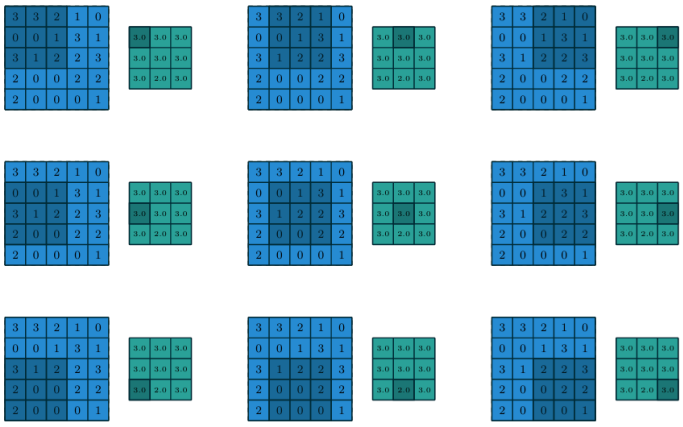

Figure 2.4: When using max pooling, each region of the input feature map is reduced to its maximum value [7].

One of the forms of discrete convolution fundamental to GANs is transposed convolution. It is in some way an inverse to convolution, making it possible to convert tensor resembling the output feature map of convolution into tensor with shape similar to its input feature map. Among all types of transposed convolution, the most important for GANs is the transposed con-

volution with non-unit strides, also named fractionally strided convolution. Figure 2.5 helps to visualize how the transposed convolution with non-unit strides is performed.

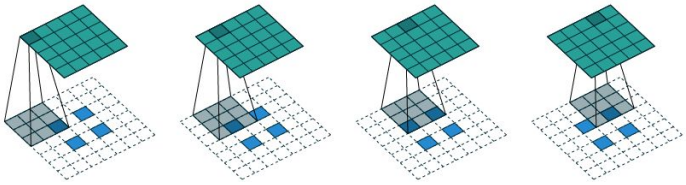

Figure 2.5: By including values among the elements of the input feature map it is feasible to force kernel to take more steps than in the case of convolution with a single stride, which leads to a larger size of the output feature map compared to the input feature map [7].

2.2 Deep convolutional generative adversarial networks

Until 2014, most convolutional neural networks were constructed using convolutional blocks with max-pooling layers in between with a small number of fully connected layers at the end of a contracting path to perform image classification [9]. This includes Alexnet and VGG16, the architecture of which is shown in the Figure 2.6. [9] investigated the effect of substituting pooling operations with convolution blocks with non-unit strides. They reached classification performance comparable with best conventional CNNs with a simple model relying exclusively on convolutions for feature extraction.

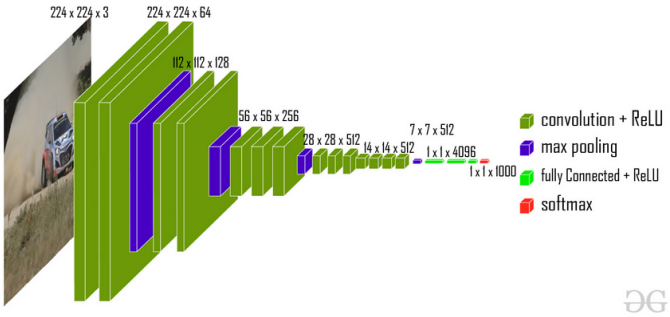

Figure 2.6: VGG-16 architecture from [10].

Inspired by the success of CNNs in computer vision and borrowing from [9] the idea of using fully convolutional network without pooling [6] developed a GAN setting, where generator and discriminator were composed of convolu-

tional layers. It was called Deep convolutional generative adversarial network (DCGAN).

The authors also experimented with reducing the number of fully connected layers, considering for example global pooling. This method was finally dropped due to the increased training time. Instead, the deepest convolutional blocks were attached directly to the initial dense layer in the generator and final decision making dense layer in the discriminator. Activations in layers were normalized by means of batch normalization suggested by [11] to reduce the potential problems coming from setting wrong initial parameters and accelerating training.

The final model was based on the following principles:

- Build discriminator with strided convolutional layers instead of pooling and generator with transposed convolutional layers
- Incorporate batch normalization into discriminator as well as generator for robust training and avoiding vanishing and exploding gradients as well as mode collapse
- Refrain from using multiple dense layers
- Choose hyperbolic tangent as an activation function for generator output layer and rectified linear unit (ReLU) activation function for all other generator layers
- Choose LeakyReLU activation function for all discriminator layers

The DCGAN generator scheme is displayed in Figure 2.7.

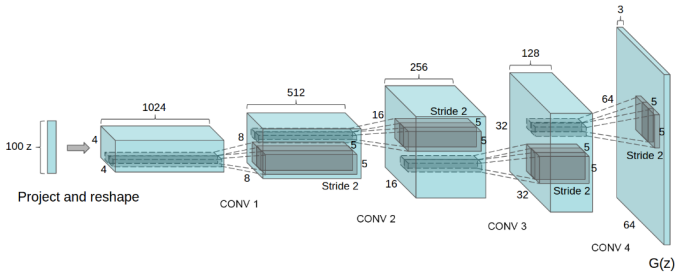

Figure 2.7: DCGAN generator for creating 64 × 64 images. The numbers above the convolutional blocks represent the convolutional kernels count.

2.3 Pix2Pix

Pix2Pix is a conditional GAN which emerged as a universal tool for supervised conversion of images from one domain to another domain [12]. The image

goes through a generator which structure resembles U-Net. Unlike U-Net, which uses upsampling to increase the size of the image in an expanding path [13], the expanding generator path is composed entirely of fractionally strided convolutional blocks. Similarly, in a contracting path, instead of max pooling blocks, convolutional layers with non-unit strides are used. Output of each layer undergoes batch normalization and ReLU nonlinearity. Skip connections between contracting and expanding path help low level features get from the input to the output. The discriminator is a PatchGAN, a model that, instead of the entire image, considers individual areas of size $N \times N$. Instead of a single value, this discriminator returns a matrix of values. Each element of the output matrix determines whether the corresponding area of the image is real or fake. It has fewer layers and thus limited set of parameters compared to a discriminator that perceives an entire image. This leads to faster learning and the ability to process bigger inputs.

The loss function comes out of the definition of the original loss function defined by [5], where the generator G takes not only a random vector z, but also an input image x, converting it into an output image y. Similarly, the discriminator receives at the input not only the resulting image, but also the initial one. Therefore, the loss function can be expressed by the following formula:

$$L_{cGAN}(G, D) = E_{x,y}[\log D(x, y)] + E_{x,z}[\log (1 - D(x, G(x, z))] \tag{2.2}$$

The authors also borrow the idea of adding per-pixel loss term to an adversarial loss [14]. This forces the generator to generate an image on the one hand that is convincing to the discriminator and on the other hand close to the target real image. L1 loss term is favored by the authors due to its capability of producing sharper images with well-defined details:

$$L_{L1}(G) = E_{x,y,z}[||y - G(x, z)||_1] \tag{2.3}$$

Thus the resulting loss function for Pix2Pix is the combination of a conditional adversarial loss and L1 loss and the optimal solution for recovering mapping between input and target images is:

$$G^* = \arg \min_G \max_D L_{cGAN}(G, D) + \lambda L_{L1}(G) \tag{2.4}$$

with λ allowing to adjust the importance of L1 loss compared to adversarial loss during training.

2.4 CycleGAN

CycleGAN allows to transform images of one style X into another style Y [15]. In order to learn corresponding transformations it does not require di-

rect mapping between training images. Instead it tries to discover the underlying mapping $G : X \rightarrow Y$ together with inverse mapping $F : Y \rightarrow X$ by taking into consideration cycle consistency $F(G(X)) \approx X$ and identity $G(Y) \approx Y, F(X) \approx X$.

2.5 FastGAN

Modern GANs are able to synthesize images of high resolution with good quality, but at the cost of large computing resources. Recently a new GAN architecture was proposed by [3], which is capable of synthesizing images with definition up to 1024×1024. According to the assurances of the authors, the model trains much faster than its counterparts and is capable of learning even on small datasets. This is possible thanks to new techniques: using skip-layer channel-wise excitation in generator and training discriminator in a self-supervised manner to encode features in order to ensure the smallest possible reconstruction error. By introducing skip-layer excitation (SLE) the authors bring the idea of skip connections to the next level. While skip connections can link only convolutional layers of the same shape SLE can be used to connect layers with different resolution. The reason in that in the SLE block addition or concatenation of output feature maps is replaced with channel-wise multiplication. The SLE block together with generator scheme is presented in Figure 2.8.

The output feature map y of the SLE with the low resolution input feature map x_{low} and high resolution input feature map x_{high} is calculated the following way:

$$y = F(x_{low}, \{W_i\}) \cdot x_{high}, \tag{2.5}$$

where W_i represents SLE block weights and F corresponds to operations applied to the low resolution feature map during SLE. At the beginning, x_{low} is downsampled by means of an adaptive average pooling, then resulting 4×4 feature map is transformed by a convolutional layer to 1×1. This feature map then goes through the LeakyReLU. After applying another convolution it gets number of channels similar to x_{high}.

Discriminator D is regularized in a self-supervised manner: it encodes features from the input images, then simple decoders try to reconstruct the original images from these features. D is coerced to seek for features that may be reconstructed better by means of reconstruction loss. In order to calculate the loss on real images feature maps f from the inner discriminator layers go through the function G and real image x from the distribution I_{real} goes through the function T. Then loss is calculated the following way:

$$L_{recons} = E_{f \sim D_{encode}(x), x \sim I_{real}}[||G(f) - T(x)||], \tag{2.6}$$

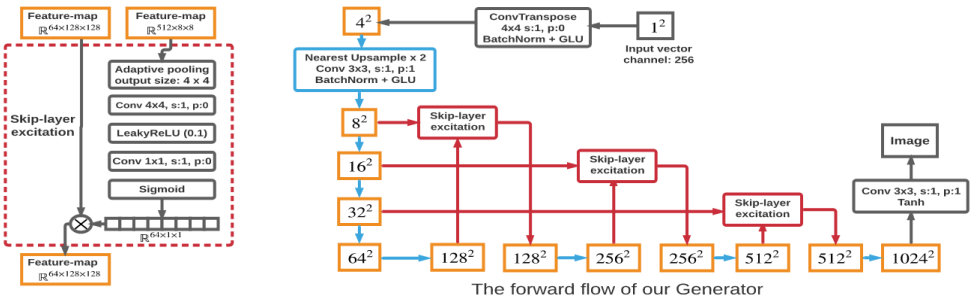

Figure 2.8: FastGAN generator. Yellow rectangles are feature maps with the number inside showing resolution of the corresponding feature map. Blue block performs upsampling.

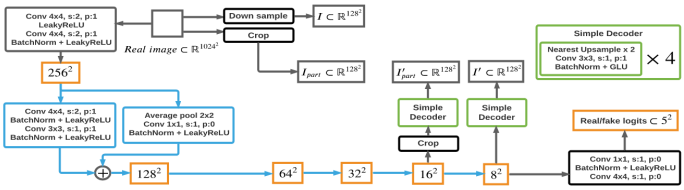

Figure 2.9: FastGAN discriminator.

2 decoders are shown In the Figure 2.9 (green rectangles): decoder f_1 if fed with images of size 16×16 and decoder f_2 is fed with images of size 8×8. These decoders with just 4 convolutional layers upscale image to size of 128 pixels. Decoder $f1$ intent is to enforce generator producing fine details. It operates on croped images of size 2×2 and outputs a reconstructed part I'_{part}, which is compared with part of the real image I_{part} cropped from the same place. Decoder f_2 helps improving general structure of the image. It is trained on encoded image and its output reconstruction I' is compared with scaled down real image I. Due to their simplicity these decoders do not bring computational overhead.

Total loss used in the FastGAN is a sum of hinge loss and reconstruction loss:

$$L_D = -E_{x \sim I_{real}}[\min(0, -1 + D(x))] - E_{\hat{x} \sim G(z)}[\min(0, -1 - D(\hat{x}))] + L_{recons} \tag{2.7}$$

$$L_G = -E_{z \sim N}[D(G(z))] \tag{2.8}$$

Hinge loss does not provide better results. It is employed due to the best computational complexity, which allows to train the model faster.

2.6 GAN evaluation metrics

Among other quantitative metrics to evaluate GAN performance [16] describes following: inception score, Frechet inception distance and multi-scale structural similarity index measure.

2.6.1 Inception score

Inception score (IS) is stated by [16] to outperform any other metric in popularity. In order to calculate IS a particular model is required: Inception Net neural network with weights learned on the ImageNet samples. The goal is to assess, whether images synthesized by the generative model may be easily separated into distinct groups one the one hand and are different from each other on the other hand. The goal is to estimate the mean KL divergence among the marginal distribution p(y) and conditional label distribution $p(y|x)$ of images. Entropy for p(y) is higher when generative model is able to produce images of each class with approximately the same probability, while low entropy indicates that the model prefers to generate images of the same class. Therefore, the goal for the generative model is to have as high entropy of $p(y)$ as possible. Meanwhile, high entropy $H(y|x)$ means, that images labeled as being of the same class are very different from each other, which is not favorable. Instead the aim for the generative model is to produce images easily identified to be related to a particular class. Thus, IS helps to measure the difference between entropies $H(y|x)$ and $H(y)$:

$$\exp E_x[KL(p(y|x)||p(y))] = \exp(H(y) - E_x[H(y|x)]) \tag{2.9}$$

Specifically, Inception network allows to determine the conditional label distribution $p(y|x)$ for sample x, while marginal distribution is estimated with:

$$p(y) \approx \frac{1}{n} \sum_{n=1}^{N} p(y|x_n = G(z_n)) \tag{2.10}$$

It was shown by [17] that higher IS is achieved by generative models that are capable of producing many varied images of better quality. This metric, however, is not without its drawbacks:

- GANs, that memorize images instead of capturing the underlying probability distribution are able to get high IS. Thus, overfitted model will no be recognized
- IS is unable to identify mode collapse
- Inception model which is necessary to calculate IS, is biased towards the ImageNet dataset it was trained on

- The value of this metric does not depend on the characteristics of the probabilistic distribution of real images P_r (therefore it possible to get high IS by generating distinct and varied images regardless of their similarity to the real images)
- IS depends on image resolution

2.6.2 Frechet inception distance

According to [18] Frechet inception distance (FID) represents the likeness of generated images and original images more precisely than IS.

Considering the distribution of generated images $p(.)$ and the distribution of original images $p_\omega(.)$ [18] introduces FID, which is represented by following equation:

$$d^2((\boldsymbol{m}, \boldsymbol{C}), (\boldsymbol{m}_\omega, \boldsymbol{C}_\omega)) = \|m - m_\omega\|_2^2 + \mathrm{Tr}(\boldsymbol{C} + \boldsymbol{C}_\omega - 2(\boldsymbol{C}\boldsymbol{C}_\omega)^{1/2}) \quad (2.11)$$

where $(\boldsymbol{m}, \boldsymbol{C})$ is Gaussian having same mean $\boldsymbol{m}$ and covariance $\boldsymbol{C}$ as the distribution of generated images and $(\boldsymbol{m}_\omega, \boldsymbol{C}_\omega)$ is Gaussian having same mean $\boldsymbol{m}_\omega$ and covariance $\boldsymbol{C}_\omega$ as the distribution of original images.

As in case of inception score one needs a CNN in order to calculate FID, because it involves calculating mean and covariance of the activations of a corresponding CNN in response to evaluated images. FID was stated to correlate well with human assessments. Unlike IS, it is able to detect mode collapse as well as disturbances in images. This qualities made FID a common metric for evaluating modern generative models.

Chapter 3

Images generation and evaluation

In this chapter datasets used in the work are presented. Goals are set for the augmentation of these datasets to solve 2 X-rays classification tasks and 2 segmentation tasks on Chest CT scans and dermoscopic images. Finally the images created by FastGAN are evaluated and compared with other models implemented in this work as well as results obtained by other researchers.

For each dataset, augmentation models are trained only on training data. For each task, at the end of training the model, 2000 images are generated, for which the IS and FID are calculated. Separate values are reported for images, segmentation masks and for combined images containing both image and its corresponding segmentation mask. This is done for easy comparison with results of existing and future studies. FID is measured between generated samples and training part of a corresponding dataset.

All computations are performed on author laptop GPU NVidia GeForce GTX 1050Ti Mobile or in Google Colaboratory and Kaggle notebooks with GPU Tesla K80.

3.1 Chest X-rays for tuberculosis detection

Dataset 1 is formed by merging Montgomery County CXR Set and Shenzhen Hospital CXR Set and constitutes of anonymised ribcage X-rays: 406 RGB images of healthy patients and 394 images of patients with clear indications of tuberculosis [19]. Images were resized to 256×256 pixels and divided into training and testing part. Training images include 366 normal and 355 abnormal X-rays. Testing images comprise of 40 normal and 39 abnormal X-rays.

For this dataset, the goal is set to determine whether augmentation will increase the accuracy of the VGG16 classifier in detecting tuberculosis.

3.1.1 CycleGAN

3 CycleGANs with perceptive fields of 46, 22 and 10 are trained for 200 epochs each to perform unsupervised image-to-image trainslation from normal to abnormal X-rays and vise versa. Then each model generates 326 abnormal X-rays from normal training samples and 316 normal X-rays from abnormal training samples. Charachteristics of the generated images are presented in the Table 3.1.

Table 3.1: CycleGAN generated tuberculosis X-rays quality

Model	Perceptive field	X-rays	Inception score	Frechet inception distance
1	46	Normal	2.25	136.55
1	46	Abnormal	2.42	133.73
2	22	Normal	2.47	143.81
2	22	Abnormal	2.54	114.94
3	10	Normal	2.39	92.98
3	10	Abnormal	2.55	91.13

The FID values indicate that the model with the discriminator with the smallest field of view of 10 pixels performs the transformation better than other models and achieves better similarity with real X-rays. It is decided to use 183 normal generated X-rays and 177 abnormal generated X-rays to extend the dataset.

3.1.2 FastGAN

2 FastGANs are trained separately on normal and abnormal X-rays. Normal FastGAN is trained for 50000 epochs and abnormal FastGAN is trained for 52000 epochs.

IS and FID for generated images may be found in Table 3.2. Generated X-rays together with real images from the dataset can be seen on Figure 3.1.

Based on good perceived quality and FID the decision is made to use 183 synthetic normal X-rays and 177 synthetic abnormal X-rays to augment the dataset for tuberculosis detection.

3.2 Chest X-rays for viral pneumonia detection

Dataset 2 was created as a joint effort of Qatar University and University of Dhaka research group together with contributors from Pakistan and Malaysia and healthcare professionals [20]. Among other samples it contains 10200 normal ribs X-rays divided into 9180 training and 1020 testing samples and 1345 X-rays of patients with viral pneumonia split into 1211 training and 134 testing samples. The dataset is therefore unbalanced due to the significant

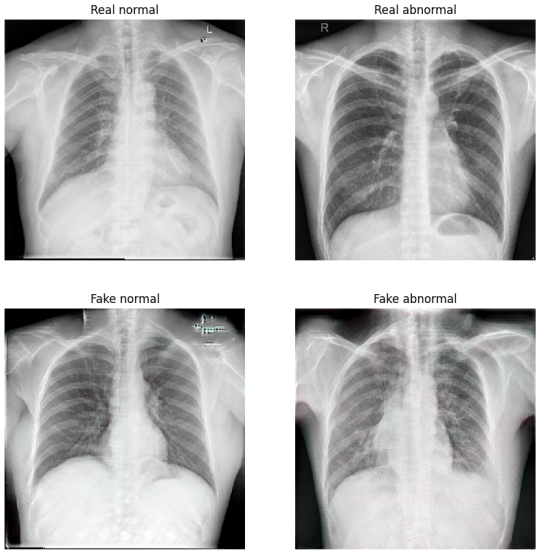

Figure 3.1: Real (top) and generated (bottom) tuberculosis detection X-rays.

preponderance of normal X-rays over X-rays with viral pneumonia indications. Images are resized to 256×256 pixels.

The goal for this dataset is to generate viral pneumonia X-rays and evaluate the effect of such augmentation on viral pneumonia detection accuracy by means of VGG16 classifier. FastGAN is trained for 58000 epochs. However, after epoch 42000 generated images perceived quality starts to drop. Model trained for 42000 is selected in order to generate X-rays for augmentation.

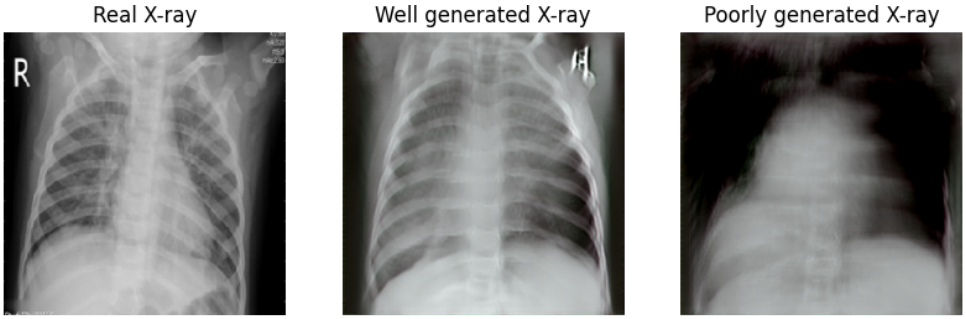

Figure 3.2: Real and generated viral pneumonia X-rays.

3.3 Chest CT scans for lung and heart segmentation

Dataset 3 is made by selecting 3100 CT scans and their corresponding segmentation masks from CT Lung and Heart and Trachea segmentation dataset, which, according to the author [21] contains the biggest number of segmentations from all chest CT segmentations datasets. Samples are resized to 256×256 pixels. 3000 CT scans are used for training and 100 scans are used for testing.

FastGAN is trained for 60000 epochs. IS and FID for generated images may be found in Table 3.2. Generated CT scans and segmentation masks together with real samples from the dataset can be seen in Figure 3.3.

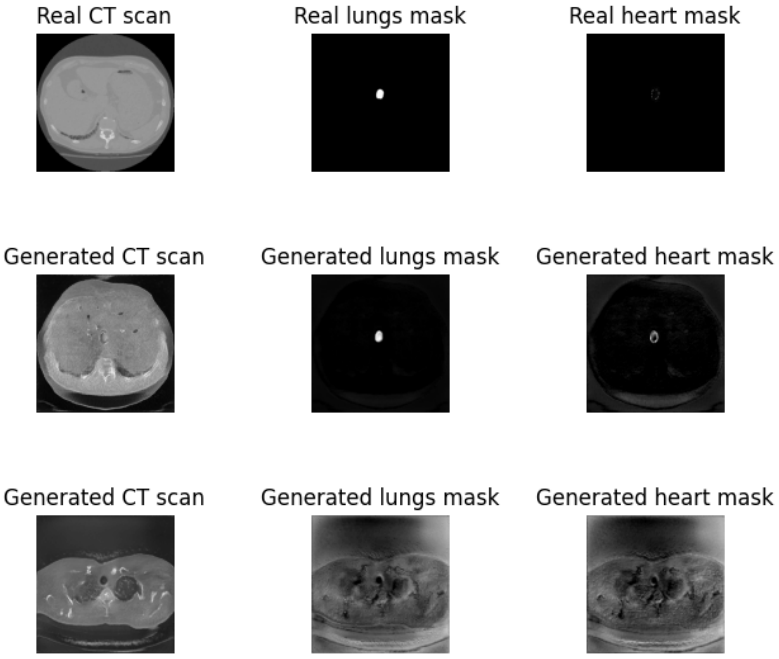

Figure 3.3: Real (top first row) and generated (second and third rows) chest CT scans. The third row shows the case where FastGAN is not able to correctly separate image channel from segmentation masks channels.

3.4 Melanoma segmentation

Dataset 4 is ISIC 2018 Skin Lesion Analysis dataset. 2594 epidermic wounds images were gathered from multiple body parts using different dermatoscopes [22]. Each image is paired with segmentation mask corresponding to the position of the wound. Images are resized to 256×256 pixels and grayscaled

Table 3.2: FastGAN generated images quality for each dataset

Dataset	Inception score	Frechet inception distance
Tuberculosis X-rays (normal)	1.58	104.99
Tuberculosis X-rays (abnormal)	1.40	143.91
Viral pneumonia X-rays	2.14	189.20
Chest CT (images)	2.66	225.58
Chest CT (lung masks)	2.89	323.73
Chest CT (heart masks)	3.59	371.52
Chest CT (combined)	2.43	183.30
Melanomas (images)	3.74	130.83
Melanomas (masks)	3.19	140.85
Melanomas (combined)	2.75	87.82

in order to reduce training time and memory requirements. Both real and fake lesions and segmentation masks are shown in Figure 3.4. It is decided to use 1000 generated images and segmentation masks to augment the dataset due to their good visual quality.

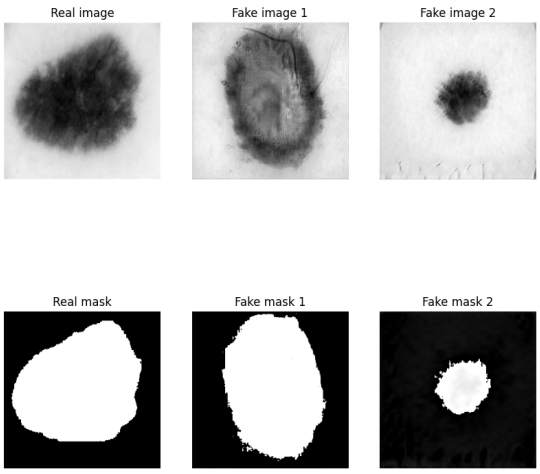

Figure 3.4: Real image and real segmentation mask (left side) and 2 generated image-segmentation mask pairs (right side). It can be seen that FastGAN learned how to add realistic bubbles to image, but is incapable of generating proper hair around the wound.

Chapter 4

Medical imaging data augmentation

In this chapter images previously generated by selected models are used for X-ray tuberculosis dataset augmentation to improve tuberculosis detection accuracy and to expand chest CT and ISIC 2018 datasets to increase segmentation accuracy of U-Net segmentation network.

4.1 Tuberculosis detection with pretrained VGG16

Keras VGG16 model is used with weights pretrained on ImageNet dataset for features extraction. Model is trained on 366 normal and 355 abnormal X-rays. 40 normal and 39 abnormal samples from the training dataset are used for validation. Model is first fine-tuned on real data for 100 epochs for each hyperparameters set with batch size of 16. All VGG16 weights remain constant during training. Adaptive moment estimation optimizer is utilized for training. Hyperameters values, validation accuracy, validation false positives and false negatives are shown in Table 4.1.

Model with the best hyperparameters is then evaluated on the test data. Test classification accuracy, false positives and false negatives are shown in Table 4.2.

Then model is trained in the same fashion on the dataset augmented with FastGAN, CycleGAN 1, CycleGAN 2 and CycleGAN 3 and results are compared. IS and FID values for each model can be seen in the Table 4.2.

From all the GAN models used in this experiment only one was able to improve the classification accuracy: CycleGAN2 with perceptive field of 22 pixels. As a result of augmentation, the classification accuracy on test data increased by 4 percent, the number of false positives decreased by 2 and the number of false negatives by 5, compared to the model trained only on real data.

Table 4.1: Validation results for fine-tuning tuberculosis detection model on real data. Rows are sorted by classification accuracy in descending order

	Learning rate	Dropout rate	Classification accuracy	False positives	False negatives
1	0.0003	0.3	0.90	8	8
2	0.0002	0.4	0.90	8	8
3	0.0002	0.5	0.90	8	9
4	0.0001	0.4	0.90	8	9
5	0.0003	0.5	0.90	9	9
6	0.0001	0.3	0.90	9	9
7	0.0002	0.3	0.89	9	9
8	0.0003	0.4	0.89	8	11
9	0.0001	0.5	0.89	9	11

Table 4.2: Test results for models trained on real and augmented datasets

Dataset	Classification accuracy	False positives	False negatives
Real	0.76	17	20
FastGAN	0.76	18	19
CycleGAN1	0.76	18	19
CycleGAN2	0.80	15	15
CycleGAN3	0.70	28	23

4.2 Melanomas segmentation

[23] showed, that it is possible to improve segmentation precision by adding GAN-generated images to the ISBI ISIC 2017 Skin Lesion Segmentation Challenge dataset. Model based on Pix2Pix architecture was used to synthesize 128×128 melanoma images from ground truth segmentation masks. However, many details of the experiment were not disclosed, for example, the number of convolutional filters and the number of layers in both generator and discriminator.

Based on their success, the desicion is made to use their model as a baseline for FastGAN. Several discriminator architectures are compared and their parameters are adjusted to identify the best perfoming model. Experiments include using perceptual loss as in [24] during training instead of L1 loss. Moreover, inspired by [25], in addition, it is shown that DCGAN can be used to generate segmentation masks for lesion segmentation, which in the case of [23] were generated manually with the addition of elastic deformations.

Two segmentation models are used to assess the effectiveness of data augmentation: Pix2Pix for 128×128 images and U-Net for 256×256 images.

Table 4.3: Test results for segmentation models trained on real and augmented datasets

Dataset	Classification accuracy	False positives	False negatives
Real			
FastGAN	0.74	535.04	619.40

4.3 Tuberculosis classification with pre-trained VGG16

Model based on pre-trained VGG16 CNN model from Keras is used to classify X-ray images into two groups: normal images and images with pneumonia. On top of pre-trained VGG16 there is batch normalization layer, then global average pooling layer, then dropout layer.

4.3.1 Image classification with VGG16

Classification model is built on top of VGG16 which was pre-trained on the ImageNet dataset containing thousands of pictures [26]. This is called transfer learning - approach which allows to overcome lack of training samples by applying models trained on other problem sets [27] as well as making training faster.

4.3.2 Training VGG16 on real Xray data

VGG16 is trained on 721 X-ray images: 366 healthy and 355 abnormal. 20% of training images are used for validation. After 100 epochs training accuracy of 0.96 and validation accuracy of 0.78 is achieved. Trained model is then tested on 79 X-rays from which 40 are healthy and 39 are abnormal. Test accuracy is 0.70. Big gap between training and validation accuracy may be caused by model overfitting on training data. In order to avoid overfitting data is augmented with three transformations:

- Random horizontal flip
- Random rotation in range from 36° clockwise to 36° counter-clockwise.
- Random zooming out and zooming by maximum amount of 10%

After adding rescaling and data augmentation training accuracy after 100 epochs became 0.82, validation accuracy 0.81 and test accuracy of 0.69. Decrease in test accuracy and increase in validation and test accuracy may be explained by augmentation helping to overcome overfitting of the model on training data. After 1000 epochs test accuracy was 0.86, validation accuracy 0.77 and test accuracy 0.71.

4.3.3 Data augmentation

4.3.3.1 DCGAN

Model is built using Tensorflow2 keras API.

Discriminator consists of following layers:

1. Convolutional 2-dimensional layer with 64 output feature maps, square kernel of size 5, strides of value 2 expecting 64×64 pixels grayscale image as an input

2. LeakyReLU acrivation function layer

3. Dropout layer with dropout rate of

4.

4.3.4 Measuring GAN performance

In order to measure GAN performance all synthesized images are classified by VGG16 trained on real images and classification accuracy is evaluated. Since separate models are created to generate healthy and abnormal images, it is known in advance that a GAN model trained on healthy images produces only images labeled healthy, and a GAN trained on abnormal radiographs creates only radiographs marked abnormal. Classification accuracy on synthesized abnormal x-rays is 0.02.

Table 4.4: Classification accuracy of VGG16 trained on real data on synthesized images

model	classification accuracy
DCGAN	0.695

4.3.5 Classification accuracy

Model was trained for 1000 epochs.

Table 4.5: Classification accuracy of VGG16

data	test accuracy
original data	0.70
original data after rescaling and augmentation	0.76

4.4 Generating melanoma images from segmentation masks

[23] showed, that it is possible to improve segmentation precision by adding GAN-generated images to the ISBI ISIC 2017 Skin Lesion Segmentation Challenge dataset. Model based on Pix2Pix architecture was used to synthesize 128×128 melanoma images from ground truth segmentation masks. However, many details of the experiment were not disclosed, for example, the number of convolutional filters and the number of layers in both generator and discriminator.

Based on their success, we complemented the experiment by comparing several discriminator architectures and varying their parameters. We also tried to use perceptual loss as in [24] during training instead of L1 loss. Moreover, inspired by [25], who introduced the idea of using multiple discriminators operating at different scales, we evaluated advantages of using 2 and 3 discriminators during training. In addition, we showed that GANs can also be used to generate segmentation masks, which in the case of [23] were generated manually with the addition of elastic deformations.

4.4.1 Dataset and preprocessing

Dataset is the ISIC 2018 Skin Lesion Analysis dataset, which contains the largest publicly available collection of quality controlled dermoscopic images of skin lesions [22]. It consists of 2694 melanoma images and the same number of ground truth segmentation masks. All images were resized to 128×128 pixels and divided into 2076 training, 518 validation and 100 test data pairs.

4.4.2 Generating segmentation masks

Ground truth segmentation masks for synthetic melanoma images were generated with DCGAN. DCGAN architecture was based on model proposed by [28]. Generator was fed with 100-dimensional random noise vector, which was then transformed to two-dimensional output by project and reshape block. Project and reshape block was followed by sequential encoder blocks. Finally convolution was applied to get output 128×128 image.

Following changes were made to overcome graphical memory restrictions: dense layer in the project and reshape block contained 8 times less units and was reshaped to 512 output feature maps of 4×4 pixels instead of 1024 feature maps with size 8×8 pixels. In this regard, there was a need for an additional decoder block. Thus generator was built using 5 transposed convolution layers with 512, 256, 128, 64 and 32 convolutional filters respectively. Transposed convolutions were followed by convolutional layer, which translated 16 input feature maps to the resulting grayscale image. Similarly one convolutional layer was added to the the discriminator. Discriminator consisted of 6 con-

volutional layers with 16, 32, 64, 128, 256 and 512 fiters respectively. All convolution and transposed convolution layers used 5x5 kernels similarly to [28]. The generator and discriminator architecture is shown in the figure 4.1.

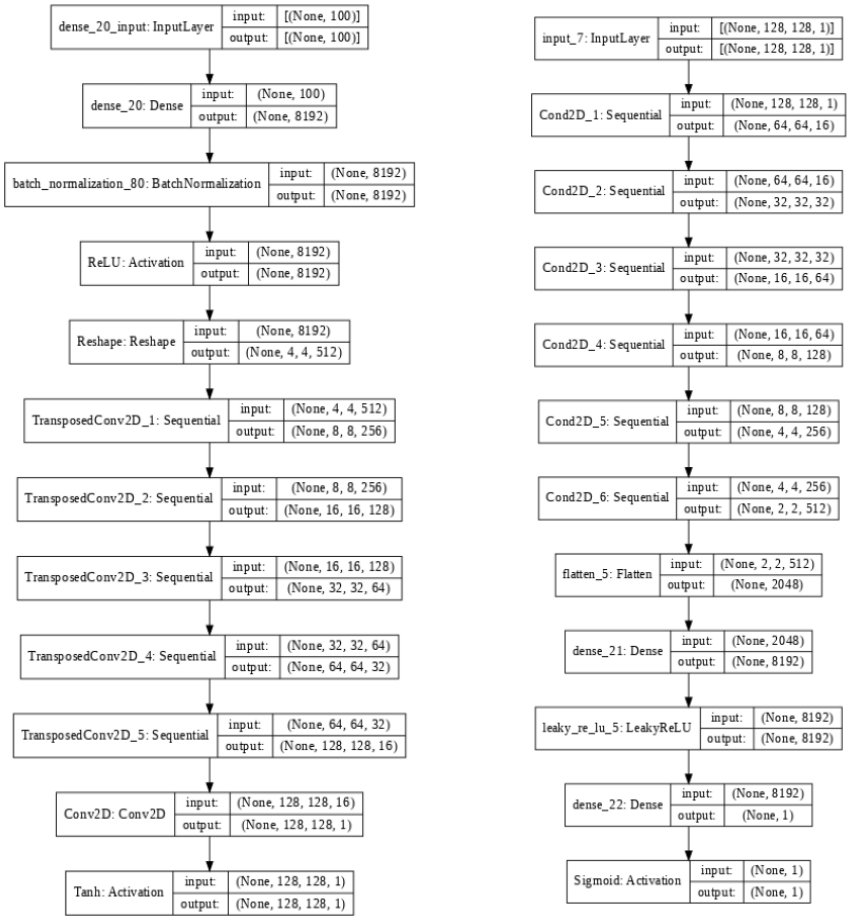

Figure 4.1: Segmentation masks DCGAN generator architecture

Original adversarial loss and adaptive moment estimation optimizer with learning rate of 0.0001 were used during training . Model was trained for 100 epochs.

To get IS and FID score 2594 segmentation masks were generated, since the size of the dataset must be large enough so that the values of these metrics correspond to the real characteristics of the synthesized images. Model achieved IS 1.86 and FID 861.41.

Then model was trained for another 100 epochs with the intention of investigating the behavior of the generator and discriminator during further training. After 200 epochs generated images had IS 2.69 and FID 1827.49.

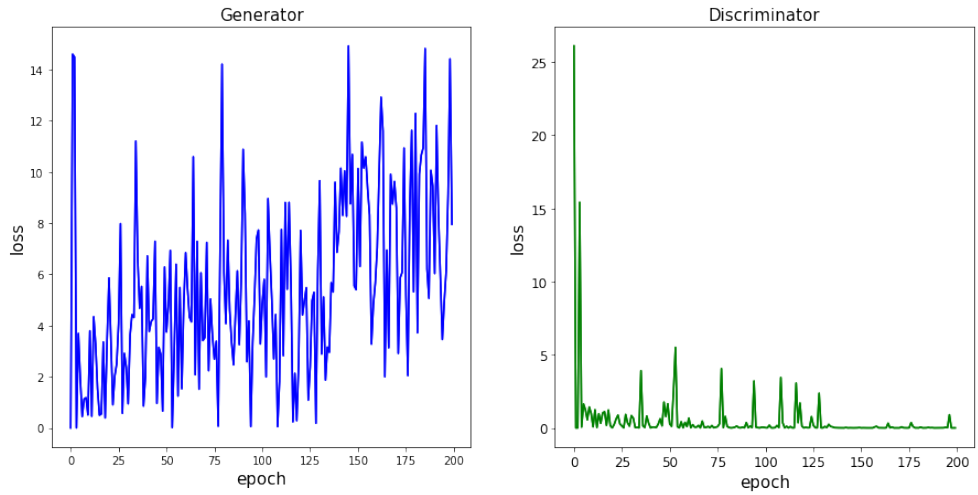

Figure 4.2: Generator and discriminator losses for segmentation masks synthesizer during 200 epochs

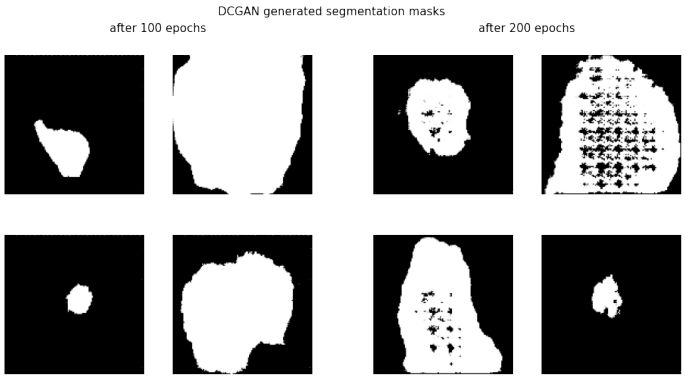

Figure 4.3: DCGAN generated segmentation masks after 100 epochs (left) and 200 epochs (right)

The generator and discriminator adversarial losses are presented in the figure 4.2. Looking at the graphs we see that the discriminator learned to distinguish real from fake images significantly faster than the generator learned to generate plausible images. Moreover, generator loss function was oscillating through the whole training process, there was no sign of conversion. As a consequence, resulting generated segmentation masks after 200 epochs were of worse quality than images generated after 100 epochs.

The substantially higher value of the FID confirmed that the statistical distribution of synthetic images after an additional 100 training epochs significantly moved away from the statistical distribution of real images. This

was another prove that relying only on classical adversarial loss often leads to GAN training being inconsistent as denoted in [29].

4.4.3 Translating segmentation masks to melanoma images

Melanoma images from segmentation masks were generated with Pix2Pix GAN. All models used U-Net-like generator consisting of the encoder and decoder with skip connections, which was proved to be efficient for image-to-image translation by [12]. Implementation was based on [30]. Because input and output resolution was 2 times smaller, than in case of [12], both encoder and decoder had 1 block less, than original generator. All other parameters including kernel size 4, strides 2 and dropout rate 0.5 were not changed. Annotation for describing generator and discriminator architecture was borrowed from [12]. 2 main blocks were used: Ck standing for convolution layer followed by batch normalization and ReLU activation function and CDk similar to Ck with dropout applied before batch normalization. The structure of the encoder and decoder in this notation was C64-C128-C256-C512-C512-C512-C512 and CD512-CD512-C512-C256-C128-C64 respectively. 3 discriminators with receptive field of 10, 22 and 46 were evaluated: Discriminator1, Discriminator2, Discriminator3 with following structures: C64-C128-C256, C64-C128 and C64.

Receptive field of each convolutional layer cell was calculated using following formula:

$$I = K + S(O - 1), \tag{4.1}$$

where I is receptive field size, K is kernel size, S is stride and O is output size along one dimension. Resulting receptive field was found starting from the last convolutional layer. In case of Discriminator 3 it was:

$$I_{Final} = 4 + 1(1 - 1) = 4 \tag{4.2}$$

$$I_{C64} = K_{C64} + S_{C64}(I_{Final} - 1) = 4 + 2(4 - 1) = 4 = 10 \tag{4.3}$$

For Discriminator2:

$$I_{Final} = 4 + 2(1 - 1) = 4 \tag{4.4}$$

$$I_{C128} = 4 + 2(4 - 1) = 10 \tag{4.5}$$

$$I_{C64} = 4 + 2(10 - 1) = 22 \tag{4.6}$$

For Discriminator3:

$$I_{Final} = 4 + 2(1 - 1) = 4 \tag{4.7}$$

$$I_{C256} = 4 + 2(4 - 1) = 10 \tag{4.8}$$

$$I_{C128} = 4 + 2(10 - 1) = 22 \tag{4.9}$$

$$I_{C64} = 4 + 2(22 - 1) = 46 \tag{4.10}$$

Each model was trained for 200 epochs as in [23].

IS and FID distance for each model generated images are shown in the table:

Table 4.6: Pix2Pix models description

Model	Discriminator	Loss
1	1	Adversarial, L1
2	2	Adversarial, L1
3	3	Adversarial, L1
4	1	Adversarial, perceptual
5	2	Adversarial, perceptual
6	3	Adversarial, perceptual

Table 4.7: Pix2Pix models scores

Model	1	2	3	4	7
IS	2.21	2.36	2.39	2.23	2.30

For training with perceptual loss VGG16 pre-trained on ImageNet dataset was utilized to extract features from generated images.

Let F_i and $\lambda_{p,i}$ denote output feature maps and weight of the i^{th} layer of the feature extractor with N layers in total. Perceptual loss may be calculated as

$$L_{Percep} = \frac{1}{N} \sum_{i=0}^{N} \lambda_{p,i}(||F_i(y) - F_i(G(x))||_d), \tag{4.11}$$

where $d = 1$ for MAE and $d = 2$ for MSE loss. In models 4, 5 and 6 $\lambda_{p,1} = 100$, $\lambda_{p,2} = 200$. In models 7, 8 and 9 $\lambda_{p,1} = 100$, $\lambda_{p,2} = 200$, $\lambda_{p,3} = 300$. For all models we use MAE, i.e. $d = 1$, based on the thesis that the MSE causes image unsharpness [12].

4.4.4 Segmentation

Segmentation model architecture was the same as the Pix2Pix generator. This architecture was chosen in order to assess its suitability for another experiment.

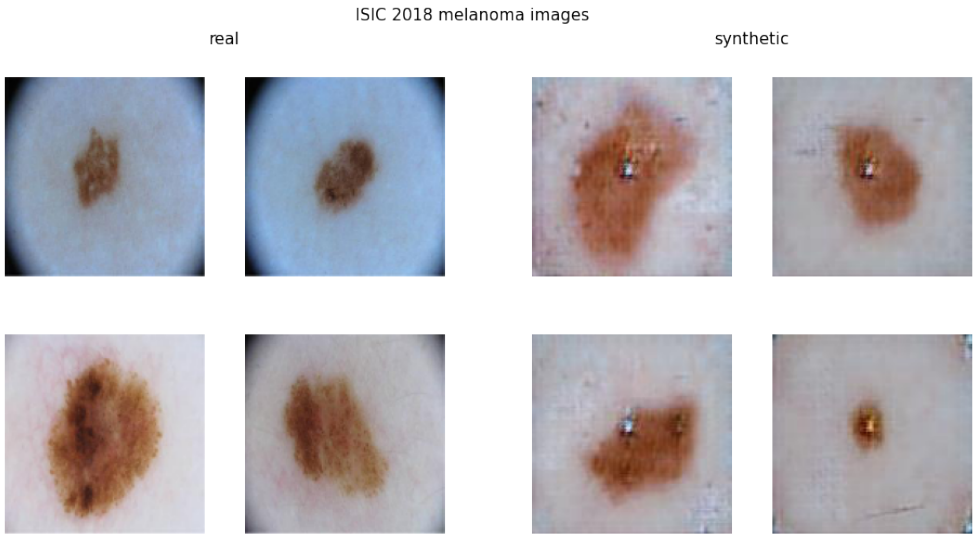

Figure 4.4: Real melanoma images (left) and Pix2Pix1 synthesized images (right)

First, the model was fine-tuned using only real samples for maximum 100 epochs for each hyperparameters set in terms of validation mean IoU. Model training stopped in case validation mean IoU not improving consistently over 15 epochs. Model trained with dropout rate 0.4 and learning rate 0.001 showed the biggest validation mean IoU of 0.83.

Table 4.8: Hyperparameters for fine-tuning melanoma segmentation model and results on validation images

	Learning rate	Dropout rate	IoU	False positives	False negatives
1	0.01	0.3	0.69	888.65	797.41
2	0.01	0.4	0.73	608.32	661.92
3	0.01	0.5	0.63	637.91	1287.27
4	0.001	0.3	0.82	352.95	464.76
5	0.001	0.4	0.83	183.77	635.96
6	0.001	0.5	0.82	305.84	521.19
7	0.0001	0.3	0.81	233.47	641.37
8	0.0001	0.4	0.81	134.67	742.74
9	0.0001	0.5	0.81	229.01	646.29

This model was then evaluated on test data showing mean IoU 0.74, mean false positives 306.74 and mean false negatives 809.98.

Second model with the similar architecture was then trained on data enriched with 1000 melanoma images and segmentation masks, generated with

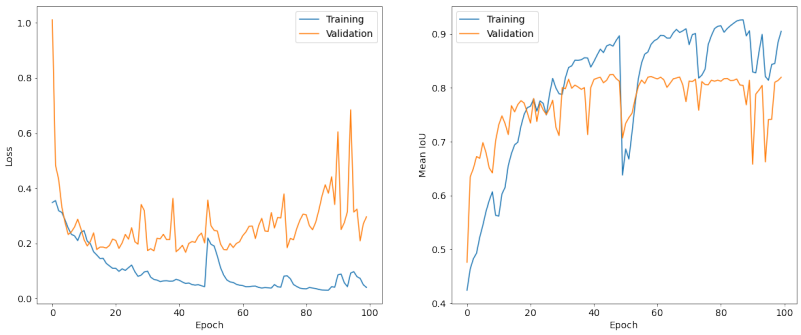

Figure 4.5: Segmentation model loss (left) and mean IoU (right) during training

simple augmentations. It gained mean IoU Third model was trained on data extended with samples generated by Pix2Pix. Results of each model are shown in the table 4.9.

Table 4.9: Segmentation results on ISIC 2018 dataset

Dataset	Mean IoU	Mean false positives	Mean false negatives
Only real	0.74	306.74	809.98
Aug simple	0.74	325.17	821.72
Aug1	0.73	342.50	790.22
Aug2	0.74	433.77	728.79
Aug3	0.74	377.44	734.69
Aug4	0.75	302.14	797.80

4.5 Small dataset augmentation

In the following experiment we research the benefits of augmentation for training segmentation model on very small dataset. The specificity of augmentation of small datasets is that the generated images may be indistinguishable from real images, but at the same time do not correspond to the segmentation masks from which they were obtained due to augmentation model overfitting.

4.5.1 Dataset and preprocessing

Dataset used for this experiment was created by selecting 240 melanoma images and ground truth segmentation masks from the ISIC 2018 Skin Lesion Analysis dataset. Images were resized to 256×256 pixels due to the limita-

tions of memory and computing resources. The dataset was divided into 160 training, 40 validation and 40 test mask-image pairs.

4.5.2 Generating segmentation masks

Segmentation masks were generated using DCGAN trained for 100 epochs.

4.5.3 Models description

Models are based on U-Net arhitecture as in the previous experiment with the only difference - additional encoder and decoder block is added to double the resolution of input and output images.

Model 1 is original Pix2Pix which is unlike original model and is trained without random jitter. Generator and discriminator architecture is same as in [12]. Model 2 has the same architecture as model 1, but is trained with random jitter.

Table 4.10: Pix2Pix scores depending on architecture

Model	1	2	3	4	5	6
IS	3.87					
FID	232.048					

Conclusion

In this work is the goal to evaluate capabilities of a newly proposed generative adversarial network architecture and to compare it with selected GAN models currently used for medical visual data augmentation.

Theoretical part gives the reader an overall understanding of methods used in this work. Chapter 1 is an introduction to machine learning terms necessary for understanding more advanced principles explained in the next chapter. Chapter 2 introduces the reader to generative adversarial models used to generate images. Chapter 3 explains what datasets and GAN models were used for images generation and quality assessment. Chapter 4 contains detailed information on experimenting with augmentation of selected datasets to improve classification and segmentation accuracy.

Although the new model FastGAN is capable of generating good quality high-resolution images, the images it generated could not improve the results of training for the classification model based on VGG16 and segmentation models based on U-Net and Pix2Pix. To improve the results, it is necessary to try to augment the data used to train generative models as well as to experiment more with models architectures and parameters including number of layers and number of convolutional filters. It could be also beneficial to try different training parameters, which include learning rate and different types of optimizers.

Bibliography

1. HAN, Changhee; HAYASHI, Hideaki; RUNDO, Leonardo; ARAKI, Ryosuke; SHIMODA, Wataru; MURAMATSU, Shinichi; FURUKAWA, Yujiro; MAURI, Giancarlo; NAKAYAMA, Hideki. GAN-based synthetic brain MR image generation. In: *2018 IEEE 15th International Symposium on Biomedical Imaging (ISBI 2018)*. 2018, pp. 734–738. Available from DOI: 10.1109/ISBI.2018.8363678.
2. MOTAMED, Saman; ROGALLA, Patrik; KHALVATI, Farzad. Data Augmentation Using Generative Adversarial Networks (GANs) For GAN-Based Detection Of Pneumonia And COVID-19 In Chest X-Ray Images. 2021.
3. LIU, Bingchen; ZHU, Yizhe; SONG, Kunpeng; ELGAMMAL, Ahmed. Towards Faster and Stabilized GAN Training for High-fidelity Few-shot Image Synthesis. *arXiv e-prints*. 2021, arXiv–2101.
4. HURWITZ, Judith; DANIEL, Kirsch. *Machine learning for dummies. Hoboken, NJ: John Wiley and Sons, Inc.* 2018. ISBN 978-1-119-45495-3.
5. GOODFELLOW, Ian J.; POUGET-ABADIE, Jean; MIRZA, Mehdi; XU, Bing; WARDE-FARLEY, David; OZAIR, Sherjil; COURVILLE, Aaron; BENGIO, Yoshua. *Generative Adversarial Networks*. 2014. Available from arXiv: 1406.2661 [stat.ML].
6. RADFORD, Alec; METZ, Luke; CHINTALA, Soumith. *Unsupervised Representation Learning with Deep Convolutional Generative Adversarial Networks*. 2016. Available from arXiv: 1511.06434 [cs.LG].
7. DUMOULIN, Vincent; VISIN, Francesco. *A guide to convolution arithmetic for deep learning*. 2018. Available from arXiv: 1603.07285 [stat.ML].
8. KRIZHEVSKY, Alex; SUTSKEVER, Ilya; HINTON, Geoffrey E. Imagenet classification with deep convolutional neural networks. *Advances in neural information processing systems*. 2012, vol. 25, pp. 1097–1105.

9. SPRINGENBERG, Jost Tobias; DOSOVITSKIY, Alexey; BROX, Thomas; RIEDMILLER, Martin. *Striving for Simplicity: The All Convolutional Net*. 2015. Available from arXiv: 1412.6806 [cs.LG].

10. *VGG-16: CNN model [online]. GeeksforGeeks.* 2020. [10.5.2021]. Available also from: https://www.geeksforgeeks.org/vgg-16-cnn-model/.

11. IOFFE, Sergey; SZEGEDY, Christian. Batch normalization: Accelerating deep network training by reducing internal covariate shift. In: *International conference on machine learning*. 2015, pp. 448–456.

12. ISOLA, Phillip; ZHU, Jun-Yan; ZHOU, Tinghui; EFROS, Alexei A. Image-to-image translation with conditional adversarial networks. In: *Proceedings of the IEEE conference on computer vision and pattern recognition*. 2017, pp. 1125–1134.

13. RONNEBERGER, Olaf; FISCHER, Philipp; BROX, Thomas. U-Net: Convolutional Networks for Biomedical Image Segmentation. In: NAVAB, Nassir; HORNEGGER, Joachim; WELLS, William M.; FRANGI, Alejandro F. (eds.). *Medical Image Computing and Computer-Assisted Intervention – MICCAI 2015*. Cham: Springer International Publishing, 2015, pp. 234–241. ISBN 978-3-319-24574-4.

14. PATHAK, Deepak; KRAHENBUHL, Philipp; DONAHUE, Jeff; DARRELL, Trevor; EFROS, Alexei A. Context encoders: Feature learning by inpainting. In: *Proceedings of the IEEE conference on computer vision and pattern recognition*. 2016, pp. 2536–2544.

15. ZHU, Jun-Yan; PARK, Taesung; ISOLA, Phillip; EFROS, Alexei A. Unpaired Image-To-Image Translation Using Cycle-Consistent Adversarial Networks. In: *Proceedings of the IEEE International Conference on Computer Vision (ICCV)*. 2017.

16. BORJI, Ali. Pros and cons of GAN evaluation measures. *Computer Vision and Image Understanding*. 2019, vol. 179, pp. 41–65. ISSN 1077-3142. Available from DOI: https://doi.org/10.1016/j.cviu.2018.10.009.

17. SALIMANS, Tim; GOODFELLOW, Ian; ZAREMBA, Wojciech; CHEUNG, Vicki; RADFORD, Alec; CHEN, Xi. Improved techniques for training gans. *arXiv preprint arXiv:1606.03498*. 2016.

18. HEUSEL, Martin; RAMSAUER, Hubert; UNTERTHINER, Thomas; NESSLER, Bernhard; KLAMBAUER, Günter; HOCHREITER, Sepp. GANs Trained by a Two Time-Scale Update Rule Converge to a Nash Equilibrium. *CoRR*. 2017, vol. abs/1706.08500. Available from arXiv: 1706.08500.

19. *LHNCBC Full Download List [online].* National Institutes of Health, 2021. [10.5.2021]. Available also from: https://lhncbc.nlm.nih.gov/LHC-downloads/downloads.html#tuberculosis-image-data-sets.

20. RAHMAN, Tawsifur. *COVID-19 Radiography Database. [online].* 2021. [9.5.2021]. Available also from: `https://www.kaggle.com/tawsifurrahman/covid19-radiography-database`.

21. KONYA, dr. *CT Lung Heart Trachea segmentation. [online].* 2020. [9.5.2021]. Available also from: `https://www.kaggle.com/sandorkonya/ct-lung-heart-trachea-segmentation`.

22. *ISIC 2018: skin lesion analysis towards melanoma detection. [online].* 2021. [6.5.2021]. Available also from: `https://challenge2018.isic-archive.com/`.

23. ABHISHEK, Kumar; HAMARNEH, Ghassan. Mask2lesion: Mask-constrained adversarial skin lesion image synthesis. In: *International Workshop on Simulation and Synthesis in Medical Imaging*. 2019, pp. 71–80.

24. JOHNSON, Justin; ALAHI, Alexandre; FEI-FEI, Li. Perceptual losses for real-time style transfer and super-resolution. In: *European conference on computer vision*. 2016, pp. 694–711.

25. WANG, Ting-Chun; LIU, Ming-Yu; ZHU, Jun-Yan; TAO, Andrew; KAUTZ, Jan; CATANZARO, Bryan. High-resolution image synthesis and semantic manipulation with conditional gans. In: *Proceedings of the IEEE conference on computer vision and pattern recognition*. 2018, pp. 8798–8807.

26. *ImageNet. [online].* 2021. [30.4.2021]. Available also from: `http://image-net.org/`.

27. TAN, Chuanqi; SUN, Fuchun; KONG, Tao; ZHANG, Wenchang; YANG, Chao; LIU, Chunfang. A Survey on Deep Transfer Learning. In: KŮRKOVÁ, Věra; MANOLOPOULOS, Yannis; HAMMER, Barbara; ILIADIS, Lazaros; MAGLOGIANNIS, Ilias (eds.). *Artificial Neural Networks and Machine Learning – ICANN 2018*. Cham: Springer International Publishing, 2018, pp. 270–279. ISBN 978-3-030-01424-7.

28. KORA VENU, Sagar; RAVULA, Sridhar. Evaluation of Deep Convolutional Generative Adversarial Networks for Data Augmentation of Chest X-ray Images. *Future Internet*. 2021, vol. 13, no. 1, p. 8.

29. ARJOVSKY, Martin; CHINTALA, Soumith; BOTTOU, Léon. Wasserstein generative adversarial networks. In: *International conference on machine learning*. 2017, pp. 214–223.

30. *Pix2Pix : TensorFlow Core*. [N.d.]. Available also from: `https://www.tensorflow.org/tutorials/generative/pix2pix`.

Appendix A

Acronyms

API Application programming interface

CNN Convolutional neural network

CT Computed tomography

DCGAN Deep convolutional generative adversarial network

FID Frechet inception distance

GAN Generative adversarial network

IOU Intersection over union

IS Inception score

MAE Mean absolute error

MSE Mean squared error

KL divergence Kullback-Leibler divergence

ReLU Rectified linear unit

RGB Red-green-blue

Appendix B

Contents of enclosed CD

- `readme.txt` ..... the file with CD contents description
- `exe` ..... the directory with executables
- `src` ..... the directory of source codes
  - `wbdcm` ..... implementation sources
  - `thesis` ..... the directory of LaTeX source codes of the thesis
- `text` ..... the thesis text directory
  - `thesis.pdf` ..... the thesis text in PDF format
  - `thesis.ps` ..... the thesis text in PS format