FACULTY
OF INFORMATION
TECHNOLOGY
CTU IN PRAGUE

Assignment of bachelor's thesis

Title:	Preprocessing of X-Ray images for COVID-19 detection Neural Networks
Student:	Tomáš Kořistka
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 detection and segmentation tasks in the medical imaging domain, and focus on X-Ray images. Implement one or more models that will work on open COVID-19 datasets available online. Compare their performance and focus on preprocessing data in order to achieve the best accuracy with chosen models. Discuss the pros and cons of the various preprocessing approaches. Publish your prototype code and make sure your results are reproducible.

Electronically approved by Ing. Karel Klouda, Ph.D. on 11 February 2021 in Prague.

Bachelor thesis

PREPROCESSING OF X-RAY IMAGES FOR COVID-19 DETECTION NEURAL NETWORKS

Tomáš Kořistka

Faculty of information technology CTU in Prague
Department of Applied Mathematics
Supervisor: Ing. Jakub Žitný
June 21, 2021

Czech Technical University in Prague
Faculty of information technology CTU in Prague

Reference to this thesis: Tomáš Kořistka. *Preprocessing of X-Ray images for COVID-19 detection Neural Networks.* Bachelor thesis. Czech Technical University in Prague, 2021.

Contents

List of Figures

List of Tables

List of code listing

I would like to heartily thank my family and friends for supporting me during my studies. I want to also thank my supervisor for his expertise and guidance, as well as persons working on the front line against the global pandemic.

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, mm dd yyyy .....

Abstrakt

Tato práce se věnuje nedostatečnému množství dat v doméně zobrazovacích metod v lékařství ve vztahu k viru covid-19, které má za dopad nedostatečně kvalitní klasifikaci přítomnosti tohoto onemocnění. Hlavním cílem je generování a porovnávání rentgenových snímků lidských plic. Práce zkoumá způsoby jak vyřešit problém malého množství snímků pozitivních případů patologie covid-19 použitím , vyhodnocováním a porovnáváním rozličných generativních modelů za cílem vytváření věrohodných obrázků vysoké kvality použitelných ve strojovém učení.

Práce srovnává modely z rodiny generativních adversariálních sítí a vyhodnocuje vliv použití takto vygenerovaných obrázků na trénovaní klasifikačních neuronových sítí.

Práce si dává za cíl posílit existující klasifikační algoritmy k lepšímu určování infekce virem covid-19 na základě rentgenových snímků plic.

Klíčová slova strojové učení, hluboké učení, neuronové sítě, GAN, konvoluční neuronové sítě, COVID-19, klasifikace rentgenových snímků, zobrazovací metody v lékařství

Abstract

This thesis deals with the insufficient volume of data in the domain of medical imaging related to the covid-19 virus, which hinders proper classification of this severe illness. The main focus is generation and comparison of X-ray images of human lungs. It explores ways to tackle the issue of small number of positive cases in the pathology of covid-19, by utilising, evaluating and comparing different generative models to increase the number of high quality and credible images usable in machine learning algorithms.

It compares models of the generative adversarial network family and assesses the impact of using images generated in this manner on the training of classification neural networks.

The thesis is aiming to bolster existing classification algorithms to perform better in determining covid-19 infection based on lung X-ray images.

Keywords Machine learning, Deep learning, neural networks, GAN, convolutional neural networks, COVID-19, X-rays classification, medical imaging

Summary

Motivation

Covid-19 is a highly contagious disease caused by the SARS-CoV-2 virus. Since patient zero in Wuhan, China in 2019, the disease has plunged the entire world into a health pandemic, disrupting the normal flow of life and costing millions of life worldwide. I have chosen this thesis as I saw the potential of its practical application, wanted to attain experience within deep learning and computer vision, as well as the possibility of contribution to the medical domain.

Goals

The goal of the thesis is to research state-of-the art techniques in medical imaging, and to provide a way to enhance existing classification algorithms by supplying credible data, boosting their performance by reducing the data disparity between positive and negative cases. Further, providing credible artificial data resolves the issue of medical privacy, ever present in the domain of machine learning in the medical domain.

Method

Existing classification networks were used, as well as a simple out-of-the-box CNN for the final classification. Various generative adversarial network models were used (such as the vanilla GAN, Wasserstein GAN or Deep convolutional GAN), a variational autoencoder and plain augmentation, which was also utilised in tandem with the aforementioned networks. Metrics, specifically Fréchet inception distance and Inception score were used to compare the individual models were used to compare the models. Models are implemented based on their respective papers, edited and customised to fit task-specific needs. The best performing model was incorporated into a classifier's training to evaluate its benefit.

Results

Out of the many models that were examined, implemented and their results compared and interpreted, the best performing one's generated data was used in conjunction with existing datasets to train a classifier network without hindering its performance, with accuracy of 93 % with no false negatives.

Structure

This thesis consists of 5 major parts. The first chapter establishes a baseline knowledge of terms and knowledge of game theory and machine learning. The second delves into generative adversarial networks and their many offshoots, which are the bulk of this thesis, followed by a dive into their foothold in the domain of medical imaging in the third chapter. Following is the description of implemented models and data operations in chapter 4. The entire thesis is then concisely summed in last, fifth chapter.

List of abbreviations

GDPR	General data protection regulation
GAN	Generative adversarial network
CGAN	Conditional generative adversarial network
WGAN	Wasserstein generative adversarial network
DCGAN	Deep convoluted generative adversarial network
PGGAN	Progressive growing generative adversarial network
LAPGAN	Laplacian Pyramid generative adversarial network
IS	Inception score
FID	Fréchet inception distance
EMD	Earth mover's distance
ReLu	Rectified linear unit
CNN	Convolutional neural network
FCN	Fully convolutional network

Chapter 1

Preliminaries

1.1 Game theory

Mathematical game theory, is a study of mathematical models, denoting a multi-agent environment, where agents' actions impact each other, regardless of whether the agents are cooperative or competitive (Russell and Norvig 2002). Game theory studies how rational, self-interested agents interact in a shared environment. Their outcomes affect the other agents within the environment (either directly, or by influencing the shared environment). Agent is any system that satisfies the following (Wooldridge and Jennings 1995):

- The system operates without direct intervention of humans. (Autonomy)
- The system perceives its environment and responds to it. (Reactivity)
- The system interacts with other agents. (Sociability)
- The system is actively working towards achieving its goals. (Proactivity)

▶ **Definition 1.1** (Constant-sum game (Borovicka 2013))**.** *Let $G = (\mathcal{N}, \mathcal{A}, u)$ be a game in normal form, then G is a constant-sum game if*

$$\exists c \in \mathbb{R} : \forall \mathbf{a} \in \mathcal{A} : \sum_{i=1}^{n} u_i(\mathbf{a}) = c.$$

A special case of a constant-sum game is the zero-sum game, where $c = 0$, i.e.

$$\forall \mathbf{a} \in \mathcal{A} : \sum_{i=1}^{n} u_i(\mathbf{a}) = 0.$$

▶ **Definition 1.2** (Game in extensive form (Borovicka 2013))**.** *A game in extensive form for n players is a tuple $(\mathcal{N}, \mathcal{A}, H, T, \chi, \rho, \sigma, u)$, such as:*

- *$\mathcal{N} = (N_1, N_2, ..., N_n)$ is a set of players,*
- *$\mathcal{A}$ is a set of possible actions,*
- *H is a set of decision nodes,*
- *T is a set of terminal nodes, such that $H \cap T = \emptyset$,*

- $\chi : H \to 2^{\mathcal{A}}$ *assigns a set of possible actions to each node,*
- $\rho : H \to \mathcal{N}$ *assigns to each non-terminal node a player on turn,*
- $\sigma : H \times \mathcal{A} \to H \cup T$*,*
- $u = (u_1, u_2, ..., u_n)$*,* $u_i : T \to \mathbb{R}$ *is a utility function of player* N_i *in terminal nodes.*

▶ **Definition 1.3** (Two player zero-sum game (Borovicka 2013))**.** *Two player zero-sum in extensive form is a game* $(\mathcal{N}, \mathcal{A}, H, T, \chi, \rho, \sigma, u)$ *where*

1. $| \mathcal{N} | = 2$*,*
2. $u = (u_1, u_2)$*,*
3. $\forall t \in T : u_1(t) + u_2(t) = 0$*.*

▶ **Definition 1.4** (Best response(Borovicka 2013))**.** *Given a game in normal form* $(\mathcal{N}, \mathcal{A}, u)$*, and an action profile* $\mathbf{a} = (a_{N_1}, a_{N_2}, ..., a_{N_n}))$*, let*

$$a_{\setminus i} = (a_{N_1}, ..., a_{N_{i-1}}, a_{N_{i+1}}, ..., a_{N_n},)$$

be an action profile consisting of actions of all players but N_i*. Then*

$$BR(a_{\setminus i}) = \underset{\hat{a}_{N_i} in A_i}{\arg\max}\, u_i((a_{N_1}, ..., a_{N_{i-1}}, \hat{a}_{N_i}, a_{N_{i+1}}, ..., a_{N_n}))$$

is the best response action of player N_i*.*

Best response of a player to a given set of actions taken by other players is the action that maximises that given action profile. This is mostly used to determine the best course of action in a game of limited visibility, where players choose actions independently from each other.

▶ **Definition 1.5** (Nash Equilibrium (Mesquita 2016))**.** *Considering a game in normal form* $(\mathcal{N}, \mathcal{A}, u)$ *and an action profile* $\mathbf{a} = (a_{N_1}, a_{N_2}, ..., a_{N_n}))$*,* $\boldsymbol{a}$ *is a Nash Equilibrium if*

$$\forall i \in (1, 2, ..., n) : a_{N_i} \in BR(a_{-i}).$$

In layman's terms, a Nash equilibrium is such an action profile, in which the action of each player is the best response. When players are in equilibrium, they will not change their action. It is possible for multiple such equilibria to exist.

1.2 Machine Learning

Machine learning is a discipline of artificial intelligence that has been gaining momentum in recent years and doesn't show signs of stopping. They key feature of machine learning is adapting and modeling new data without need of explicit programming, i.e. the programme creates and adapts its model(s) of its own volition. Whilst machine learning algorithms require large quantities of data and a fair bit of computational power, they can provide state of the art results and, as mentioned before, do not require a human intervention to adapt to new data. As such, they can continuously learn on their own, as more data becomes available. Machine learning algorithms comprises supervised and unsupervised learning[1].

[1] Not to be confused with the current educational tendencies.

1.2.1 Supervised learning

Supervised learning consists of inferring a mapping function of a set of inputs (a vector of features) to an output label (a supervisory signal). The model is trained on dataset of examples, and the model's performance and accuracy can be easily determined by comparing acquired results with expected values. Supervised learning can be further split into classification and regression, based on the nature of the supervisory signal[2] (Alzubi, Nayyar, and Kumar 2018).

1.2.2 Unsupervised learning

Compared to supervised learning, unsupervised learning lacks a clue as of what it is trying to achieve. Instead of finding a formula for predicting an output (label) based on data inputs (features), unsupervised machine learning aims to understand the structure, hidden patterns and data grouping (Education 2020). Clustering (cluster analysis) and anomaly detection are the most common representatives of this domain.

1.2.3 Deep learning

Deep learning is a branch of machine learning dealing with artificial neural networks. As most sources state, neural networks take their inspiration from the organic nervous system, and as such the computation cells (neurons) are connected by their synapses (inputs and outputs) that fire (represented by the output function) when their potential reaches a specific value (bias or threshold). It is important to note that deep learning can be both supervised and unsupervised.

1.2.3.1 Artificial neuron

Artificial neuron is a mathematical function consisting of a number of weights $\boldsymbol{w} = (w_1, ..., w_n)^T$, inputs $\boldsymbol{x} = (x_1, ..., x_n)^T$, a bias value x_0, for which $w_0 = 1$, an inner potential function ξ and an activation function $f(\xi)$. The set of weights of the neuron are the component which can learn in the model. Although the inner potential is almost always a sum of the weighted inputs, there are alternatives, as described by Klaus Debes 2005. The activation function for a neuron in a non-output layer is usually one of the following:

- Sigmoid: $f(\xi) = \frac{1}{1+e^{-\xi}}$ (Weisstein 2021c)
- Rectified linear unit: $f(\xi) = \max(0, \xi) = \begin{cases} \xi & \text{if} \xi \geq 0, \\ 0 & \text{otherwise} \end{cases}$ (Vasata and Cepek 2020)
- Leaky rectified linear unit: $f(\xi) = \begin{cases} \xi & \text{if} \xi \geq 0, \\ \xi * \alpha & \text{otherwise} \end{cases}$ (Maas, Hannun, and Ng 2013)
- Hyperbolic tangent: $f(\xi) = \tanh(\xi) = \frac{e^{\xi}-e^{-\xi}}{e^{\xi}+e^{-\xi}}$ (Weisstein 2021b),

whilst the output layer's function is dependent on the task:

- Regression: $f(\xi) = \xi$
- Classification to c classes[3] (Vasata and Cepek 2020): The output layer is composed of c neurons, each predicting the probability of the given class. The activation function of the

[2] For classification, the supervisory signal is a discrete value (i.e. the possible values are finite) and for regression the signal is a continuous value.

[3] Special case $c = 2$ is called a binary classification.

i-th neuron (with inner potential ξ_i) is a softmax function and represents the probability of ξ belonging to class i:

$$f_i(\xi) = \frac{e^{\xi_i}}{e^{\xi_1} + ... + e^{\xi_c}} = \hat{P}(Y = i \mid X = x).$$

Altogether the prediction for a vector of inputs x is as follows:

$$\hat{Y} = \underset{i \in 1,...,c}{\arg\max}\, \hat{P}(Y = i \mid X = x).$$

1.2.3.2 Artificial neural network

Neural networks weave individual neurons together into layers, which in turn are structured into a large, interconnected network in sequence. Each layer takes its inputs from the previous layer [4], and symmetrically feeds its outputs to the next layer[5].

Neural networks operate in two fashions - the forward and backward pass. The forward pass is the sequential calculation from inputs of the first layer to the output of the last layer, given the weights of each individual neuron. Working in the opposite direction, the backward pass calculates the error of the forward pass. The forward pass is utilised to train and predict results based on the provided inputs, as opposed to the backward pass, which is not utilised during training only.

For a given neuron, the weights are adjusted followingly (Vasata and Cepek 2020):

$$error = Y - \hat{Y}$$

$$w_i \leftarrow w_i + error * x_i$$

$$w_0 \leftarrow w_0 + error,$$

where $\hat{Y}$ is the neuron's prediction, Y the actual value, $(x_1, x_2, ...x_n)$ input values, $(w_1, w_2, ..., w_n)$ weights.

[4] Except for the very first layer, which processes the provided input data instead.
[5] Again, the last layer is an exception, and provides the result of the network to the user instead.

Chapter 2

Generative adversarial networks

2.1 GAN

Generative adversarial networks are an up and coming machine learning technique. They were first proposed by Goodfellow et al. 2014. To tackle the common issues of deep adversarial networks, a two agent model is introduced. These two agents, a generator and a discriminator, are pitted against each other in a non-cooperative game.

While the discriminator is aiming at perfectly telling apart real and fake (generated, synthetic) data, the generator's goal is to generate data such that the discriminator cannot distinguish it from real data. The two players are most usually (also described as such in the original paper) multi-layer perceptron.

Formally put, to learn the generator's distribution pg over data x, we define a prior on input noise variables $p_z(z)$, then represent a mapping to data space as $G(z;\theta_g)$, where G is a differentiable function represented by a multi-layer perceptron with parameters θ_g (Goodfellow et al. 2014). The discriminator $D(x;\theta_d)$ outputs a single scalar, where $D(x)$ is the probability that x comes from the data rather than being the output of a generator.

The utility of the generator and discriminator is to minimize $\log(1 - D(G(z)))$ and to maximize the probability of correctly labeling (distinguishing) real and fake data, i.e. $\log(D(x))$, respectively. Summarised, D and G play the following two-player minimax game with value function[1] (Goodfellow et al. 2014):

$$\min_G \max_D V(D,G) = \mathbb{E}_{x\sim p_{data}(x)}[\log(D(x))] + \mathbb{E}_{z\sim p_z(z)}[\log(1 - D(G(z)))].$$

Training then consists of a gradient descent of both networks. Goodfellow et al. 2014 used

[1] This is also referred to as the adversarial loss function.

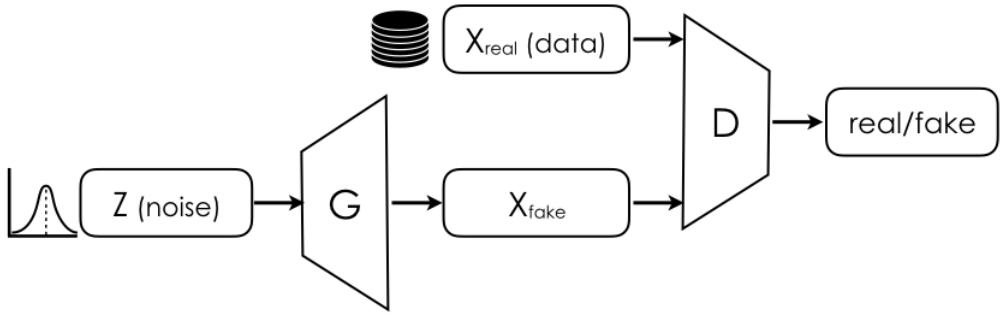

Figure 2.1 GAN schema Korkinof et al. 2019

$k = 1$ for their experiments.

Algorithm 1: Mini-batch stochastic gradient descent training of generative adversarial nets. The number of steps to apply to the discriminator, k, is a hyper-parameter.

for *number of steps* **do**
  for *k steps* **do**
    Sample mini-batch of m noise samples $(z^{(1)}, z^{(2)}, ..., z^{(m)})$ from noise prior $p_g(z)$.
    Sample mini-batch of m examples $(x^{(1)}, x^{(2)}, ..., x^{(m)})$ from data generating distribution $p_{data}(x)$.
    Update the discriminator by ascending its stochastic gradient:

$$\nabla_{\theta_d} \frac{1}{m} \sum_{i=1}^{m} [\log D(x^{(i)}) + \log (1 - D(G(z^{(i)})))].$$

  end
  Sample mini-batch of m noise samples $(z^{(1)}, z^{(2)}, ..., z^{(m)})$ from noise prior $p_g(z)$.
  Update the generator by descending its stochastic gradient:

$$\nabla_{\theta_g} \frac{1}{m} \sum_{i=1}^{m} \log (1 - D(G(z^{(i)}))).$$

end

The entire pipeline of a GAN model can be depicted by 2.1. Noise is drawn from the Z distribution, fed to the generator, which generates data X_{fake}. The discriminator is provided with both real data X_{real} and generated X_{fake} and attempts to distinguish between them. This process is laced with descents of individual networks, as mentioned by algorithm 1.

Per game theory, the entire model, composed of aforementioned two players playing a zero-sum game, converges when the discriminator D and generator G reach a Nash equilibrium, which might not exist (Farnia and Ozdaglar 2020), i.e. the model fails to converge (Salimans et al. 2016).

The main disadvantages of this approach are the need of synchronisation between the generator and discriminator, otherwise the likelihood of mode collapse – scenario, where too many values of z collapse to the same x, resulting in insufficient diversity and modality of generated data.

Another main problem that has not been resolved is the instability of model training.

The intuitive use of gradient descent to minimise each player's respective cost function results in many games not converging, and a range of heuristics have been proposed (by Salimans et al. 2016), such as feature matching, mini-batch discrimination or historical averaging.

Since its inception in 2014, the number of GAN variation has grown steeply, as depicted by 2.2, from general architectures, such as DCGAN (Radford, Metz, and Chintala 2016) or WGAN (Arjovsky, Chintala, and Bottou 2017), architectures translating data between two domains, such as pix2pix (Isola et al. 2018), CycleGAN (Zhu et al. 2020), to specialised ones like MedGAN (Armanious et al. 2020).

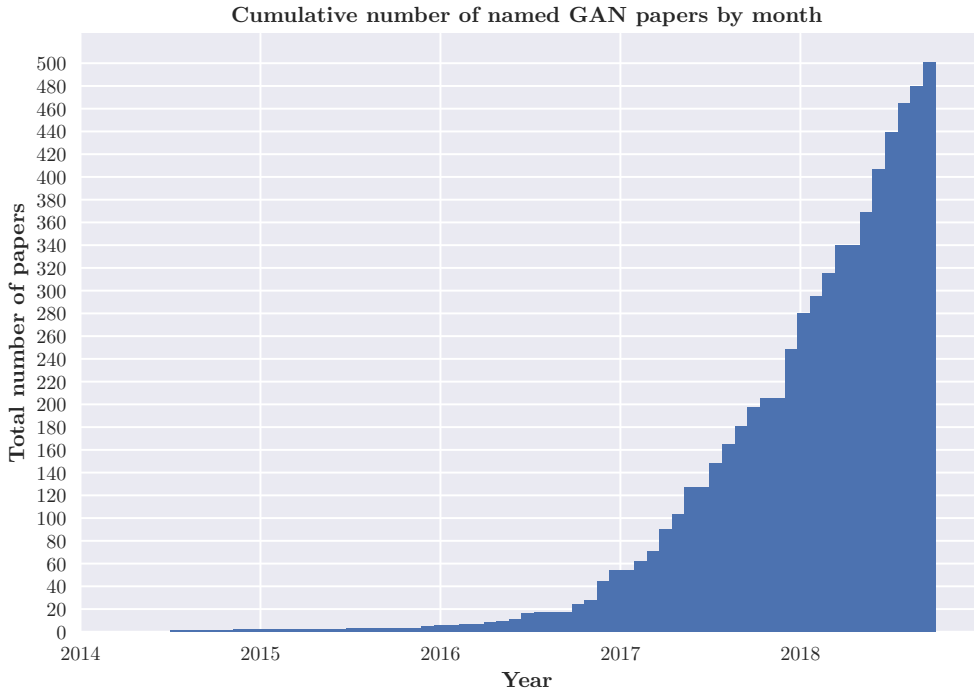

Figure 2.2 Cumulative number of named GAN papers by month. Data source: Hindupur 2017

2.2 GAN metrics

To evaluate the quality and potential application of a model as a whole, a wide range of metrics can be calculated. Whilst the individual player's utility function do provide information of their performance (such as loss or accuracy), they provide that information in the context of their adversary, they do not offer an objective view on the generated data - its variety, quality or even credibility. The two models can get caught in local extremas, and drag each other in a self-feeding loop of seemingly good performance, but producing degraded data.

Metrics measuring these qualities are aplenty - as composed by Borji 2018, the list is rather extensive (see appendix A). These techniques do not include a manual inspection (i.e. a human inspecting generated pictures and determining the quality of the result), which can quickly and cheaply identify aberrant data. However, it cannot encompass properties such as data disparity or credibility of generated data.

Most commonly used are the **Inception score** and **Fréchet inception distance**, which do precisely that - measuring generated data's variety and similarity to the real data's distribution, respectively.

2.2.1 Inception score

IS uses an existing, already trained network Inception Net (trained on the ImageNet dataset) (Szegedy et al. 2014). It is also important to note that the inception score is limited to specifically scoring images (Barratt and Sharma 2018), which is the case of this work. The goal of the inception score is **diversity** with respects to the class labels, and clear, sharp objects in the generated pictures.

▶ **Definition 2.1** (Inception score (Borji 2018))**.**

$$\exp(\mathbb{E}_x[\mathbb{KL}(p(y \mid x) \mid\mid p(y))]),$$

where $p(y \mid x)$ is the conditional label distribution for image x estimated using a pre-trained Inception model, $p(y)$ is the marginal distribution

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

and $\mathbb{KL}$ denotes the Kullback-Leibler divergence (i.e. relative entropy)[2]*.*

2.2.2 Fréchet inception distance

▶ **Definition 2.2** (Fréchet distance (Efrat et al. 2002))**.** *Let α, β be two polylines and let $d(p, q)$ denote the Euclidean distance between two points p and q in the plane. The Fréchet distance between α and β is*

$$\mathcal{F}(\alpha, \beta) = \min_{f:[0,1]\to\alpha, g:[0,1]\to\beta} \max_t d(f(t), g(t)),$$

where f and g are continuous non-decreasing functions.

Fréchet inception distance (FID) is a metric that represents distance between feature vectors of real and fake data.

2.3 Variants

2.3.1 Wasserstein GAN

To address some of the downfalls of the vanilla GAN, Arjovsky, Chintala, and Bottou 2017 propose replacing the discriminator with a critic, and using the Wasserstein distance to compare real and fake data distributions.

▶ **Definition 2.3** (Metric (Weisstein 2021c))**.** *A metric is any non-negative function $f : X \times X \to \mathbb{R}$ that satisfies the following conditions for $\forall x, y, z \in X$:*

1. *$f(x, y) = f(y, x)$ (symmetry)*
2. *$f(x, y) = 0 \leftrightarrow x = y$ (identity)*
3. *$f(x, z) + f(z, y) \geq f(x, z)$ (triangle inequality).*

Earth mover's distance [3] is a method of calculating dissimilarity between two multi-dimensional distributions. Most commonly, it is abstracted to two piles of dirt, representing the probability distributions, and the amount work needed to transform one pile into another, which is analogous to distributions' distance.

[2] The Kullback-Leibler divergence is a measure of difference between two probability distributions (Theodoridis 2020)

[3] also known as Wasserstein distance or optimal transport

▶ **Definition 2.4** (Wasserstein's distance (Arjovsky, Chintala, and Bottou 2017)). *Wasserstein metric between cumulative distribution functions F, G is defined as*

$$W(\mathbb{P}_r, \mathbb{P}_g) = \inf_{\gamma \in \prod(\mathbb{P}_r, \mathbb{P}_g)} \mathbb{E}_{(x,y)\sim\gamma}[\|x - y\|]$$

where $\prod(\mathbb{P}_r, \mathbb{P}_g)$ denotes the set of all joint distributions $\gamma(x, y)$ whose marginals are respectively Pr and Pg.

Given two probability distributions, a **transport plan** is any shifting of the weights (probabilities) of one distribution to transform it into another distribution. Taking into consideration all of these possible transport plans, the earth mover distance takes into consideration the best one, i.e. the infimum.

Algorithm 2: WGAN (Arjovsky, Chintala, and Bottou 2017)

Require:
α - learning rate; c - clipping parameter; m - batch size, n_{critic} - number of iterations of the critic per generator iteration.
Require:
w_0 - initial critic parameters; θ_0 - initial generator's parameters.
while *θ has not converged* **do**
  for *t = 0, ... n_{critic}* **do**
    Sample $\{x^{(i)}\}_{i=1}^{m} \sim \mathbb{P}_r$ a batch from real data.
    Sample $\{z^{(i)}\}_{i=1}^{m} \sim p(z)$ a batch of prior samples.
    $g_w \leftarrow \nabla_w[\frac{1}{m}\sum_{i=1}^{m} f_w(x^{(i)}) - \frac{1}{m}\sum_{i=1}^{m} f_w(g_\theta(z^{(i)}))]$
    $w \leftarrow w + \alpha \cdot RMSProp(w, g_w)$
    $w \leftarrow clip(w, -c, c)$
  Sample $\{z^{(i)}\}_{i=1}^{m} \sim p(z)$ a batch of prior samples.
  $g_\theta \leftarrow -\nabla_\theta \frac{1}{m}\sum_{i=1}^{m} f_w(g_\theta(z^{(i)}))$
  $\theta \leftarrow \theta - \alpha \cdot RMSProp(\theta, g_\theta)$

The main advantage of utilising a Wasserstein distance is that it is continuous and differentiable and has a linear gradient - it is possible to train until optimum without a mode collapse. The discriminator is replaced by a critic, who acts as a helper for estimating EMD between real and generated data distributions.

Instead of predicting the probability of data being real or fake[4], the critic scores the 'realness' or 'fakeness' of data. The critic, when trained to optimum, it functions as as a loss to the generator, which can continue training, compared to the vanilla GAN which required training in tandem. A clear correlation between critic and image quality has been observed by Arjovsky, Chintala, and Bottou 2017.

Another important aspect of the WGAN model is weight clipping. In short, it is a technique that deals with exploding and vanishing gradients, i.e. the gradients are either too large or too small respectively, hindering further learning. More formally, this enforces a constraint called the Lipschitz constraint. A Lipschitz continuous function is any such function $f : \Omega \subset \mathbb{R}^{\kappa} \to \mathbb{R}$ and there exists a constant $K \geq 0$ such that:

$$|f(x) - f(y)| \leq K|x - y|,$$

for each $x, y, \in \Omega$. The important aspect of a Lipschitz continuous function is that is differentiable in almost any point.

[4] as was the case of the 'vanilla' GAN

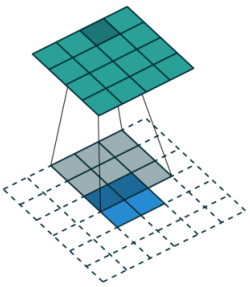

Figure 2.3 Single strided convolution of a 2×2 input with a 4×4 filter. Source: Dumoulin and Visin 2016.

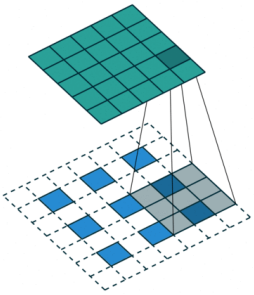

Figure 2.4 Fractionally strided convolution of a 3×3 input with stride size of 0.5. Source: Dumoulin and Visin 2016.

2.3.2 Conditional GAN

Generative adversarial nets can be extended to a conditional model if both the generator and discriminator are conditioned on some extra information y (Mirza and Osindero 2014). This extra information is provided as an input to both players. The objective function is then modified:

$$\min_G \max_D V(D,G) = \mathbb{E}_{x \sim p_{data}(x)}[\log(D(x|y))] + \mathbb{E}_{z \sim p_z(z)}[\log(1 - D(G(z|y)))].$$

Conditional GAN models (in their base form) create a one-to-one mapping of input to output, i.e. mapping of a unit of data to a single class. However, this can be naturally extended to provide a one-to-many mapping, by mapping a unit of data to class allegiance probability.

2.3.3 Deep convolutional GAN

Convolution is the process of overlaying two functions (Weisstein 2021a). In terms of images and convolutional neural networks, one of the functions is the image itself, perceived as a matrix of values, and the other is a predefined matrix, called the convolutional kernel. The kernel is applied to a section of the image, then shifted. The shift can be by a single cell, but also can be both larger and smaller, dependent on the stride size. Strided convolutions are convolutions such that the stride (step size) is larger than 1, whilst fractionally strided convolutions have a step size less than 1, which is achieved by inserting padding in-between cells to achieve the desired fraction (Dumoulin and Visin 2016).

DCGAN architecture, proposed by Radford, Metz, and Chintala 2016, uses a fully convolutional network as its generator by replacing pooling functions[5] with convolutional filters.

2.4 Variational Autoencoder

Autoencoders are unsupervised deep learning models that enforce a condensation of information by introducing a bottleneck. This way, autoencoders aim to learn the structure of the data, provided there are correlations between the input features. Essentially, they're a compression algorithm, forcing to learn a representation of data in lower dimension space.

The model consists of two transitions $\phi : \mathcal{X} \to \mathcal{F}$ and $\psi : \mathcal{F} \to \mathcal{X}$, such that

$$\phi, \psi = \arg\min_{\phi,\psi} \|\mathcal{X} - (\psi \circ \phi)\mathcal{X}\|^2 .$$

[5] Also called down sampling layers, which force the network to concentrate important features into a data structure of smaller size.

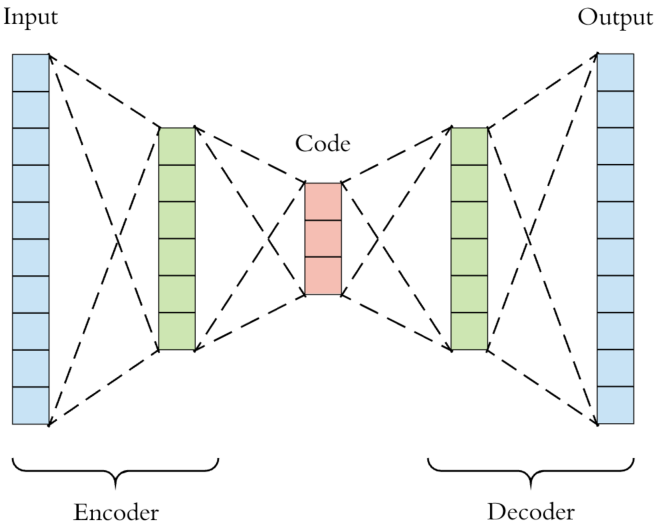

Figure 2.5 Autoencoder general architecture. The input and output layers are identical in size, and the hidden layers (code) derive the latent representation of the data. Source: Stewart 2019.

Variational auntoencoders venture off this path by instead of representing the bottleneck with a vector of latent attributes, representing it by probability distributions. The encoder is approximating a posterior distribution, formulating it as an optimization task. The goal is to generate data x from latent variables z. i.e. learning:

$$p(z \mid x) = \frac{p(x \mid z)p(z)}{p(x)},$$

where

$$p(x) = \sum_{z} p(x \mid z)p(z),$$

which could be intractable, and as a result, the posterior distribution[6] is intractable. To solve this debacle, commonly the $p(z \mid x)$ is approximated by a different distribution $q(z \mid x)$, such that it is tractable and at the same time is not too different to the former. To minimise the difference between the two distributions, the Kullback-Leibler (KL) divergence (Joyce 2011) is used.

$$D_{KL}(p||q) = \sum_{i=1}^{N} p(x_i) \log p(x_i) - \log q(x_i) = \sum_{i=1}^{N} p(x_i) \log \frac{p(x_i)}{q(x_i)},$$

which is very similar to the entropy of a probability distribution (Shannon 1948):

$$H = -\sum_{i=1}^{N} p(x_i) \log p(x_i).$$

From that focus, the KL divergence is easily interpreted as the expected loss of information by substituting the p distribution with the q distribution.

Put altogether, the expression that the variational autoencoder attempts to minimise is:

$$E_{q(z|x)} \log p(x \mid z) - KL(q(z \mid x)||p(z))$$

[6]The summary **after** after the data has been observed - new evidence.

Essentially, the network learns by learning the mean and standard deviation of the latent space. Training data that is similar (i.e. the same class label) stimulates similar nerves (passes the threshold of the same neurons) and forms clusters in the latent space (Stewart 2019).

Chapter 3

Medical imaging

Medical imaging is used to examine tissue and organs[1] in-vivo[2]. It is an umbrella term for a numerous collection of techniques and processes used to create images of various modalities of the human[3] body.

This includes, but is not limited to (Mayo et al. 2018 and *Imaging and radiology* 1997):

1. X-rays are a type of electromagnetic radiation. Their most important property is that they are absorbed by dense tissues, such are bones, or contrasting agents injected into bodies, and pass through soft issue without much difficulty. Commonly, x-ray examinations are used to diagnose bone damage and fractures, but can also be used to examine parts of the gastrointestinal tract, such as the stomach or the intestines, and other internal organs, such as the heart, the bladder or lungs (*X-Rays* 2021; *Imaging and radiology* 1997).

2. Magnetic resonance imaging (MRI) exploits interaction of magnetic fields and water molecules' alignment in the body. The procedure is carried out by two powerful magnets, first of which causes the hydrogen atoms to align in one direction within the body, whilst the second is rapidly turned on and off. As different kinds of body tissues react differently to this pull, and as such can be observed and segmented (*MRI scan* 2018).

3. Sonography uses high frequency sound waves released by probes (transducers), which also act as receivers. Based on the delay between the transmission and the reflection of the sound wave, the distance can be calculated and an image produced (Mental Health 2016).

4. Positron emission tomography (PET) traces the radioactive emission of an injected tracer agent. This can map metabolic processes, blood flow, oxygen and glucose consumption (Berger 2003). These observations are used to evaluate the functionality (or dysfunctionality) of organs and tissues, such as cancer, heart disease or neurological disorders (*Positron Emission Tomography - Computed Tomography (PET/CT)* 2019).

5. Computer tomography (CT) is a procedure that utilises x-rays to create a three dimensional model of the target. In a fashion similar to how a gamma knife (*Gamma Knife* 2018) operates, a CT procedure consists of narrow x-ray beams rotated around the body, creating slices of the body, which can be stacked to construct a three dimensional representation (Mental Health 2013).

[1] And basically everything that should stay on the inside of the human meat puppet
[2] 'In the living body of a plant or an animal'. (Merriam-Webster 2021)
[3] Not necessarily limited to human anatomy, though.

Each of the different imaging approaches provides a different kind of information, and as such is used for different diagnoses, and different modalities are often utilised in tandem in a cross modality evaluation to complement each other, resulting in a better diagnosis.

3.1 The quality dilemma

The problem with image data, to which medical images are no exception, is that it is incredibly disparate in regards to its properties, such as image resolution, image size, number of colour channels, never mind the actual semantic information of the content. This poses a daunting challenge, as deep learning algorithms require a single, unified format of input data. As a result, these properties should be perceived as a hyper-parameter, and treated accordingly.

Focusing on resolution alone, there is a number of trade-offs. Larger resolution provides more information[4] and allow training to be more accurate. Armanious et al. 2020 state that 'especially when using automated image analysis tools, high image quality is required for the accuracy and reliability of the results', which supports the point of view of containing as much relevant information as possible per image.

However, image resolution goes hand-in-hand with the number of parameters to be optimised. As a result, high resolution images explode the size of the learning models, and increases the affinity of overfitting (Mesquita 2016).

Although unintuitive, empirically it shows that models work better when trained on lower resolution images. This discussion is still at large, but one thing is certain - high resolution images, especially in medical imaging, are a resource sought after. Resolution (and number of color channels, etc.) is easier to scale down rather than up - resulting in information loss at worst, compared to facilitating artefacts in the image.

Another important factor is the actual need for detail. As examined and shown by Sabottke and Spieler 2020, many important findings are clearly noticeable at fairly low resolutions. In particular, in their study of chest radiography Sabottke and Spieler 2020 have examined images ranging from 32×32 (pixels) to 600×600, and found that best results were achieved for resolutions between 256×256 and 448×448 for a range of binary classifications. As seen in figure 3.1 (retrieved from Sabottke and Spieler 2020), findings of the pathology are visible even at low resolutions.

The main advantage of using low resolution data is increase in computation, but relaxing the resolution requirement also allows more data to be used, resulting in larger, bulkier and (hopefully) diverse datasets.

The effects of a covid-19 pneumonia are, as most pneumoniæ, visible on x-ray modality images. The telltale indicator is an increased density in the lungs, visible as (in layman's terms) whiteness in the lungs' x-rays. The more severe the pneumonia, the more opaque the white, effectively decreasing the visibility of the common lung marking.

3.2 The quantity conundrum

The biggest concern when it comes to machine learning is data quantity. As the aim of machine learning is to model the general realities based on a limited sample, the larger the dataset that encompasses rich, various data[5], the better. Smaller datasets can easily lead to a model's overfitting, producing pointless data of no use, just like uniform datasets[6].

Data with high class imbalance (a large disparity in the volume of data per class) leads to poor classification. This is the case in the domain of medical imaging, for a number of reasons:

[4] Whether the information is relevant or a dead-end is to be determined.
[5] That ideally represents the entire domain.
[6] Dataset without much variance.

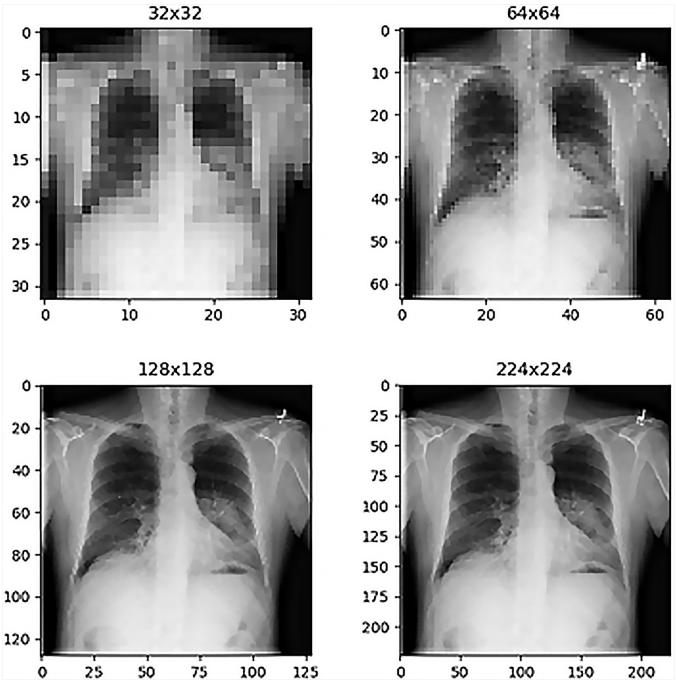

Figure 3.1 Comparison of chest radiographs at different image resolutions for patient 103 (60-year-old man with a thoracic mass). The mass finding is visible in all images but with visually observable improved clarity in the higher resolution examples (bottom row). Source: Sabottke and Spieler 2020.

- Annotated (labeled) medical image data is sparse and expensive - it requires a medical professional's expertise to correctly annotate, and as such is time consuming and tedious[7].
- Due to moral values (patients' privacy and confidentiality[8]), medical images can be difficult to access to the general public. Each country handles this problem differently by laws, and there is no single accord all follow, but in general, any data regarding medical patients has to be anonymised prior to their release. In USA, this falls under 'The Privacy rule' federal law (*Your Rights Under HIPAA* 2020), this is covered by GDPR in EU.
- The number of data for a given pathology (positive cases) are scarce (Yi, Walia, and Babyn 2019), despite the effort of organisations such as RSNA or or NBIA. In other words, data depicting abnormal findings are not nearly as numerous as normal findings.

With GDPR in effect, health data is treated specially as sensitive in nature by default, and can be processed only if (Aumage 2016)

- an individual's consent is given to do so, or
- it is necessary for archiving purposes, or
- it is used for scientific, historical or statistic purposes.

o

[7] Also, a lot of medical data is often three dimensional (Shin, Roth, et al. 2016; Guibas, Virdi, and P. S. Li 2018).

[8] Handling private information of patients in confidence by medical professionals.

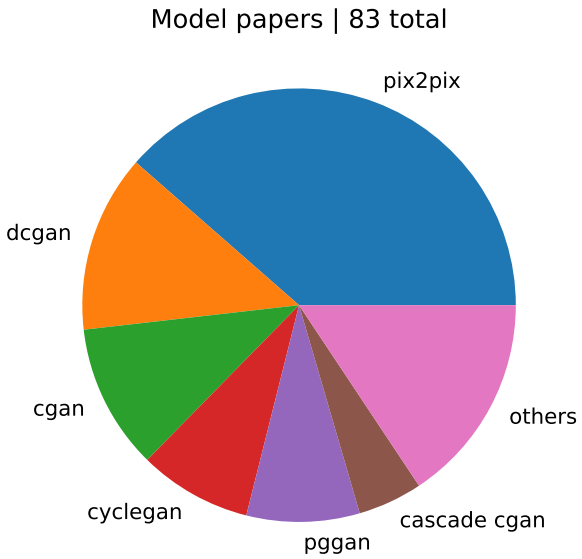

Figure 3.2 Distribution of the GAN-based models utilised in medical imaging. Note that this examines the medical imaging domain as a whole, and as such is not telling of individual sub-domains (such as image synthesis or translation). Data source: composite of Yi, Walia, and Babyn 2019 and Singh and Raza 2020.

3.3 State of the art

Deep learning has established a firm foothold in the domain of medical imaging. Multiple parties have compiled lists of notable GAN-based models in medical imaging (Yi, Walia, and Babyn 2019 and Singh and Raza 2020). Figure 3.2 shows the distribution of different models in the domain of medical imaging, with pix2pix being the most common, and DCGAN following.

3.3.1 Reconstruction

Yu et al. 2017 improves the speed of MR imaging, patient discomfort and reduces the suspectibility to artefacts induced by patients' movement by undersampling and then using a combination of a U-Net architecture in the generator and a trio of losses - adversarial, a pixel-wise mean squared error and a perceptual loss. 1708.00961 uses similar methods, but omits the pixel-wise MSE due to it being prone to generating blurry images.

Abramian and Eklund 2019 set out to an unprecedented goal - reconstruction of facial features from anonymised MRI. They were examining if the state of the art anonymisation techniques were sufficient, using a CycleGAN, with a large degree of success in terms of face blurring, and limited success in complete face removal reconstruction.

Shan et al. 2018 commits transfer learning from 2D trained networks to 3D to take into account information from adjacent MR slices during reconstruction. This is then taken into account with convolutional filters shifting over neighbouring slices.

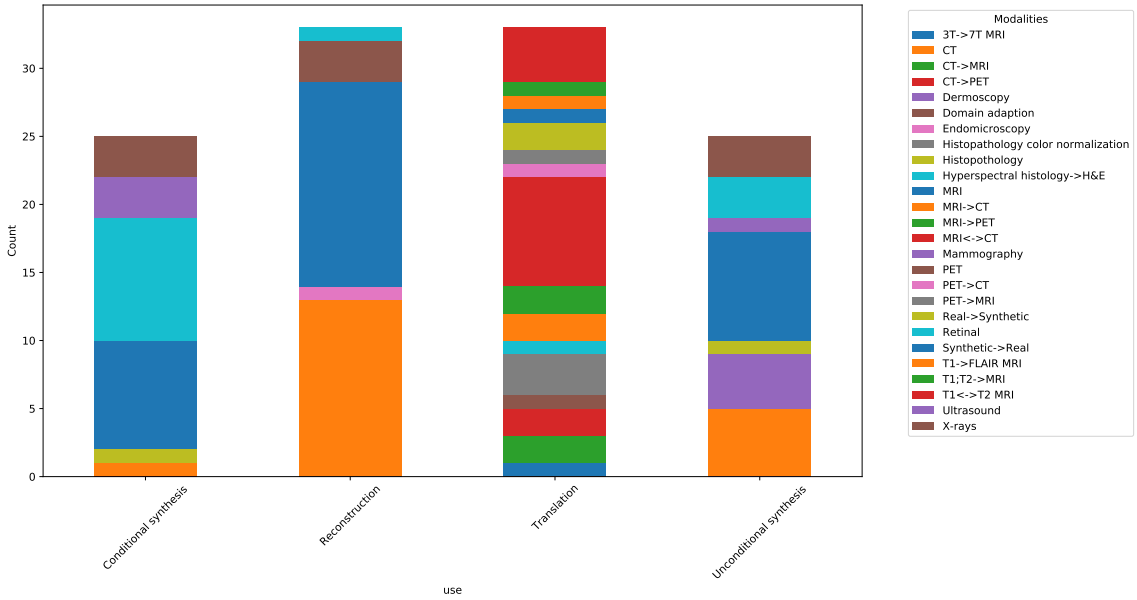

Figure 3.3 Distribution of GAN models in different domains. Entries containing an arrow depict a translation from a domain to another domain. Data source: composite of Yi, Walia, and Babyn 2019 and Singh and Raza 2020.

3.3.2 Synthesis

Synthesis is, alongside translation, the center stage of GANs in medical imaging. Almost every application of a GAN model results is one of a kind modification of an existing model[9].

Yi and Babyn 2018 used a DCGAN to synthesise image of lung nodules, and instead of computed metrics, used a visual Turing test to determine the quality of the images. Baur, Albarqouni, and Navab 2018b and Baur, Albarqouni, and Navab 2018a worked with skin lesion data, the former with a LAPGAN modified to work with a single source of noise and upsampling instead of interpolation, whilst the later used a progressive growing GAN (PGGAN). Similarly, Beers et al. 2018 used a PGGAN to produce MR images of retinal fundus[10], Bowles et al. 2018 also added Gaussian noise layer (blur) based on empiric results to further stabilise training. Mok and Chung 2019 used two generators, first of which trained to sketch a coarse shape and texture of a brain MR image, whilst the second refines it to a high resolution, detailed image. Calimeri et al. 2017 made use of LAPGAN in the same domain, Han et al. 2018 of WGAN.

Possibly the highest resolution images in medical imaging were synthesised with the use of a Wasserstein GAN by Korkinof et al. 2019, who worked with more than one million mammogram images, and much like the previously mentioned, progressively increased the resolution during the training, up to 1280×1024. Notably, they have also used FID and IS to evaluate their models. Mahapatra and Bozorgtabar 2019 tackles the issue of high resolution by chaining GAN networks with a triplet loss[11].

Zhao, H. Li, and Cheng 2017, Iqbal and Ali 2018, Appan K. and Sivaswamy 2018, Costa et al. 2018 and Guibas, Virdi, and P. S. Li 2018 proposed various models to generate retinal fundus images, with the first one also applying the same approach to neuronal images. Iqbal and Ali 2018 and Zhao, H. Li, and Cheng 2017 both utilised style transfer to boost training, and the former also updated generator's values twice as often to reduce training time.

[9] Todo what is visual Turing test

[10] The inside lining of an eyeball (Stöppler 2021).

[11] Comparing the generated data with 'truth-y' and 'false-y' data (Das 2019).

A frequent use of GAN models is synthesis of Brain MRI. Plassard et al. 2018 used a slice of MRI, whilst Shin, Tenenholtz, et al. 2018 uses labels (masks of tumours), enforcing variability by altering the properties of the label, such as size or placement, or even placing it over a tumour-free image.

Chuquicusma et al. 2018 proposed using a DCGAN to model the underlying distribution of lung cancer CT images and compared the results using a visual Turing test, whereas Salehinejad et al. 2018 made a more generic sweep by exploiting to boost the number of various pathologies captured by X-ray images of chest, training the model of a dataset of 2000 images a class. Madani et al. 2018 addresses the issue using a semi-supervised learning approach, training on both labeled and unlabeled data. Even though the unlabeled data does not provide information as of its class, it functions similarly to a transfer style training - X-ray images of lungs will follow a general shape and proportions, in spite of positive sickness or damage. This technique was also applied to retinal fundus imaging by Lahiri et al. 2018.

There exist a set of guidelines for good cardiac magnetic resonance images, and to that end, Zhang, Gooya, and Frangi 2017 aimed to automate that assessment. As in most cases though, the volume of training data is sparse, and so additional data is generated by a DCGAN.

3.3.3 Translation

Translation is the process of mapping between different modalities of the same underlying image. The motivation behind translation is replacing expensive or dangerous procedures with more appropriate ones, by finding correlations between different modalities, removing artefacts, i.e. any information that is introduced by the machine/process (Bell 2020, overcoming imaging constraints such as lack of a contrast medium or the amount of a radiation dose the patient is exposed to during the procedure.

Most of the research in translation to CT is focused on brain images, such as Emami et al. 2018, Nie, Xiang, et al. 2019, Wolterink et al. 2017, Nie, Trullo, et al. 2016. Emami et al. 2018's generator constitutes of a ResNet with shortcuts between distant layers, which offers conservation of lower level details. Wolterink et al. 2017 feeds unpaired data to a CycleGAN.

Whilst brain imaging is consistent as to image shapes, orientation and imaging angles, the same is not true for many domains. Mahapatra, Bozorgtabar, et al. 2019 works with this variety with hips, Hiasa et al. 2018 and Maspero et al. 2018 utilised a CycleGAN and pix2pix to translate between MR and CT pelvic images, aiming to overcome MR's poor contrast for bone analysis. Working in the opposite direction, Jin et al. 2019 made use of both paired and unpaired data of brain images, employing two loss cycle consistencies.

Ben-Cohen et al. 2017, Bi et al. 2017 mapped CT to PET. The former used a CGAN and a fully convolutional network (FCN). The CGAN's results were more realistic, but the FCN responded to malignant tumour. The authors blended the two models by placing a mask from the FCN onto images made by CGAN.

Medical image translation can also take place within the same modality, with different properties. For instance, whilst higher doses of radiation are potentially more harmful, they yield better results than lower doses. As such, there is a notion to map low dose CTs to high dose CTs, as done by Yi and Babyn 2018, Liu et al. 2020; Nie, Xiang, et al. 2019, first of which added a third neural network to the equation, addressing the general problem of edge blurring, whilst the last deploys two discriminator. One of them functions on local areas of images, and the other on the global scale of the image.

MRI is capable of a similar mapping. Based on magnetic resonance weights [12] MR images depict different types of tissue. Nie, Xiang, et al. 2019 added a second discriminator to the model's architecture. While one discriminator examines the images in their global scope, the second works on a local one to address the sway of easily synthesised regions (specifically, healthy tissue).

[12] How much time elapses between a proton's excitation and relaxation (Preston 2006)

A peculiar case is the work of Mahmood, Chen, and Durr 2018, who had approached model learning in an inverse manner. The generator is replaced with a transformer, which learns to map real data to synthetic data instead.

Armanious et al. 2020 set out to create a universal image translation model for medical imaging. They introduce a handful of novelty components, such as two loss functions, or a generator comprised of a chain of encoder-decoder networks, to progressively grow the desired images.

3.3.4 Segmentation

Segmentation has been mentioned in previous sections, but only its results were used. Image segmentation is the process of creating a pixel-wise mask over the original image, identifying a group of pixels that represent an object. In medical imaging the most common use of image segmentation is identification of abnormal regions or alien bodies, such as tumours, or boundaries of organs. Xue et al. 2018 drew inspiration from GANs, manifesting in an adversarial model that consists of a segmentation network and a critic with a custom loss for the segmentation network, which is an encoder-decoder in the original paper with skip connections between corresponding layers. The model has been trained to segment brain tumours with a superior accuracy.

As with the other domains, three dimensional data is an open problem, mainly due to the tedious amount of work needed to label all slices (most of which contains information already provided in adjacent slices), and as such, properly annotated three dimensional data is sparse. Currently, there are groups that work with this issue (Çiçek et al. 2016, Milletari, Navab, and Ahmadi 2016), all of which train their models on either images or worst case large sections, rather than small patches. Çiçek et al. 2016 segments brain tumours, Milletari, Navab, and Ahmadi 2016 segments prostate in MRI and adds a dice similarity loss function, Dou et al. 2016 learns to segment livers from medical images. Yang et al. 2017 segments liaisons in 3D CT data, producing a probability map, again, using an encoder-decoder as a generator.

A name that has been mentioned multiple times is project U-Net Ronneberger, Fischer, and Brox 2015. U-net is a model that aims to create a rich, deep convolutional segmentation network by ideally assigning a class label to each element of the picture. It sports a U-shape architecture - consists of two parts, the first of which is constraining and progressively reducing the number of layers, and the other expanding, working in the opposite direction - a concept similar to autoencoders. U-Net has demonstrated fantastic results in medical imaging, and serves as a baseline for many other works.

Dai et al. 2017 streamlines shape and irregularity based diagnosis of lungs and heart by performing segmentation on chest x-rays. The model is made of a segmentation network, and a critic, both being fully convolutional networks.

Chapter 4

Implementation

4.1 Technologies

Experiments were conducted using Python. The main motivation behind that is, aside from the relative ease and comfort of writing code in Python, the popularity it has attained in the domain of machine learning, data analysis and data operations. The specific distribution of Python can differ on project basis, but Python 3 is the foothold of the implementation, varying from versions 3.5 to 3.8.

As this project falls in the field of computer vision, the computing power of a desktop station level computer is not sufficient. Instead, data operations and transformations, model learning and testing were all run in cloud to make use of more powerful hardware. Namely, models were trained in Google Colab and Deepnote environments. Both environments' preferred modus operandi are interactive Jupyter notebooks documents[1]

Most image datasets are accompanied by a metadata `csv` file, containing supplementary information to the image data. To that end, Pandas team 2020 a data manipulation library, was used to read, edit and save data.

There are multiple frameworks for working with artificial neural networks. The most widely used ones are Tensorflow (Developers 2021), Keras (Chollet et al. 2015), Caffe (Jia et al. 2014) and PyTorch (Paszke et al. 2019). Tensorflow has two main distributions (1.0 and 2.0), both of which are used at some point during experiments, and which differ in their API. To make matters worse, whilst Tensorflow 2.0 provides legacy support to Tensorflow 1.0, it still requires changes in the user code to run. Google Colab provides an advantage in that regard, easily allowing a switch between the two versions by running the in-built magic command 4.1.

```
1 % tensorflow_version 1.x
```

Code listing 4.1 Tensorflow version change in Google Colab from the default 2.0 to 1.0 for legacy code.

```
import wandb

...

wandb.init(project='GAN',
    group='GAN',
    entity='koristo1',
    config={
        'epochs'      : epochs,
```

[1]Deepnote offers a terminal, therefore `.py` scripts could be run directly.

```
        'batch_size' : batch_size,
        'height'     : self.img_rows,
        'width'      : self.img_cols,
        'latent_dim' : self.latent_dim,
        'model_name' : self.timestamp,
        'model_type' : self.name
    })

...
```

Code listing 4.2 Manual configuration of the Weights and Biases listener with hyperparameters of a vanilla GAN model.

To observe and log the data, rather than creating a custom logger (or using an existing one) to store the information locally, the Weights and Biases Biewald 2020 was plugged into a project. The API provides callbacks that can be incorporated into various training and fit functions of the aforementioned frameworks, but due to the architecture complexity and irregularity of GAN models this is not feasible and the logging implementation needed to be provided manually[2], whereas the (architecturally) simpler classifiers can make full use of the provided callbacks.

```
import wandb

...

wandb.log({
    'accuracy': ...,
    'd_loss'  : ...,
    'g_loss'  : ...,
})

...
```

Code listing 4.3 Manual logging during the training process of a vanilla GAN model. Logged variables were replaced with elipsis' to simplify the snippet.

4.2 Data

Data is the be-all and end-all of machine learning. Data used in this project is provided by Wang 2020, which in turns gathers it from multiple sources. This project operates with data aggregation and selection implemented in commit `a8de16a` . Since then, the project has been expanded with more data sources. The classification models have been trained on a different dataset, provided by the implementation of Minaee et al. 2020.

Although this repository provides an easy-to-follow instructions to construct the dataset, it also produces a broken metadata file. A significant subset of both train (7.644 %) and test (8.5 %) data were corrupted this way.

4.2.1 Dataset issues

The repository does not contain the data itself, as those are curated by other parties, but it does provide direct links to the individual datasets. However, one of them, a dataset hosted on Kaggle , is broken, and the only functional source of it is now hosted on Google Drive . The dataset sources are listed in appendix B.

There are multiple issues with the provided dataset rendering it unusable without fixing them. They are as follows:

- Broken `csv` file

[2] see code snippets 4.2 and 4.3

- Disjoint number of colour channels (greyscale, RGB and RGBA)
- Different image dimension.

Fortunately, the images themselves are not corrupted as provided, and require no pruning.

The first shortcoming of the script provided by Wang 2020 is the disparity between the default names of the composing datasets and the paths the scripts assumes them to have. This is resolved by renaming the folders to mirror the expected names in the script.

As mentioned before, the dataset constructed from the Wang 2020 repository provides malfunctional data, and as such needs is to be either discarded or repaired. Meddling with the provided code is volatile, as fixing one naming issue would break another. As such, the dataset is repaired after construction. Fortunately, the error is consistent - all broken files came to be by containing a whitespace character '␣', which is also (unfortunately) used as a separator of the `csv` file.

The script 4.4 fixes this issue, by supplying an extra column which will be `NaN` for correctly formatted records, and will not be `NaN` for broken files. That is the result of the extra whitespace in the ID of the files, which consumes a column following it.

index	ID	file	label	source	err
0	5	ARDSSevere.png	negative	cohen	
60	COVID-00026	COVID-00026.jpg	positive	fig1	
67	COVID-00036	COVID-00036.jpg	positive	fig1	
104	COVID	72	COVID(72).png	positive	sirm
105	COVID	77	COVID(77).png	positive	sirm
109	COVID	95	COVID(95).png	positive	sirm
116	COVID	213	COVID(213).png	positive	sirm
119	COVID	216	COVID(216).png	positive	sirm
389	284	000011-6.jpg	negative	cohen	
483	75	covid-19-pneumonia-19.jpg	positive	cohen	

Table 4.1 Examples of unbroken and broken data entries. The broken entries have a value in the `err` column.

The table 4.1 shows an example of five non-corrupted and five corrupted entries, respectively. The `ID` of the corrupted entries is split, half of it occupying the `file` column, forcing the rest of the columns to be shifted as well. The script below fixes that issue, by providing an artificial `err` column, which is `NaN` for non-corrupted entries, whereas the corrupted entries occupy the column (due to the `ID` spanning two columns instead of one). Based on the value of this extra column, the dataset can be split and regenerated. The broken entries have their `ID` concatenated with the column following it, followed by dropping the (now) extra column, and renaming the columns to be in accordance with their data.

```
import pandas as pd
from PIL import Image

def verify(df, subset):
  broken_files = []

  for file in df['file']:
    filename = './data/{}/{}'.format(subset, file)

    try:
      Image.open(filename)
    except FileNotFoundError:
      broken_files.append(filename)

  print('No. of broken files: {}'.format(len(broken_files)))
```

```
16   print('Files: {}'.format(broken_files))
17
18   return len(broken_files) == 0
19
20 def fix(name):
21   df = pd.read_csv('{}_split.txt'.format(name),names=['id','file','label', '
       other','err'],sep=' ')
22
23   broken = df[ df['err'].notna()].copy(deep = True)
24   broken['id'] = broken['id'] + '(' + broken['file'] + ')'
25
26   cols = broken.columns[:-1]
27   broken = broken.drop(columns=['file'])
28   broken.columns = cols
29
30   df = df.drop(columns=['err'])
31   df = df.drop(broken.index)
32
33   result = pd.concat([df,broken])
34   if verify(result, name):
35     result.to_csv('{}_split_fixed.txt'.format(name), sep = ' ', header = None)
```

Code listing 4.4 Dataset fixing

Observing the same entries as before, the fixed dataset is represented by table 4.2.This dataset is usable to work over the provided images.

index	ID	file	label	source
0	5	ARDSSevere.png	negative	cohen
60	COVID-00026	COVID-00026.jpg	positive	fig1
67	COVID-00036	COVID-00036.jpg	positive	fig1
104	COVID(72)	COVID(72).png	positive	sirm
105	COVID(77)	COVID(77).png	positive	sirm
109	COVID(95)	COVID(95).png	positive	sirm
116	COVID(213)	COVID(213).png	positive	sirm
119	COVID(216)	COVID(216).png	positive	sirm
389	284	000011-6.jpg	negative	cohen
483	75	covid-19-pneumonia-19.jpg	positive	cohen

Table 4.2 Fixed dataset example

4.2.2 Data distribution

The data is comprised of covid-19 positive and covid-19 negative cases x-ray images of various dimensions and number of colour channels. As seen in table 4.3, the dataset is heavily imbalanced and skewed towards negative cases.

	train	test
positive	1670	100
negative	13794	100

Table 4.3 Number of positive and negative cases for the training and testing sets of Wang, Lin, and Wong 2020

The box and whisker plots (figure 4.1 for height and figure 4.2) are quite similar, with the interquartile range being in the 800–2000 range and being skewed upwards. Both the height and

width sit at a 1016 median value, and mean value of 1024 and 1024 respectively[3].

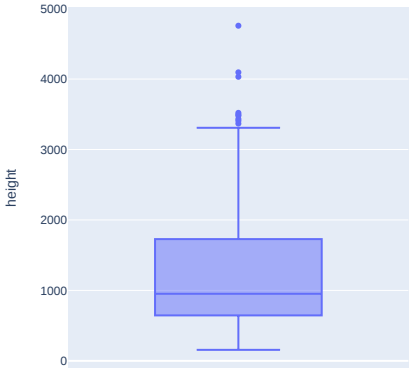

Figure 4.1 Dataset's height distribution.

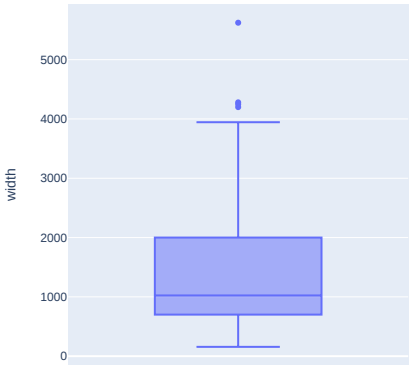

Figure 4.2 Dataset's width distribution.

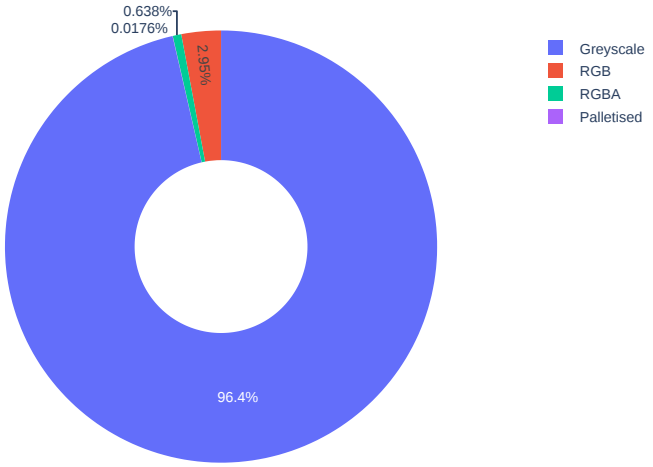

Figure 4.3 Original dataset's mode type (channel) distribution. 'Greyscale' and 'Palletised' are single channel formats, where 'Palletised' is a colour image width in single channel instead of three.

Most of the dataset (96.5 %) is already in shades of gray, whilst the remaining (3.5 %) are colour images, in one (up to 256 different colours), three or four colour channels. All non-greyscale images have been converted into greyscale images, computing the single channel pixel values as a weighted with the following weights: $0.2989, 0.5870, 0.1140$ for red, green and blue colour channels respectively.

[3] rounded to nearest whole number.

4.2.3 Augmentation

Figure 4.4 Sample of positive cases

In terms of augmenting the dataset, there are three questions that require answering.

1. What sort of transformations will be used?
2. Do the transformations create credible data?
3. Does an augmented dataset yield better results?

All three questions will be answered by evaluating the performance of models. Transformations can include any combination of affine transformations, cropping, equalisations or blurs (and many others). To preserve the variety of the original dataset, an equal number of augmented images is extracted from each (positive case) image.

The augmented dataset is created once, by using the `ImageDataGenerator` provided by `tensorflow.keras.preprocessing.image` (documentation can be found here: Chollet et al. 2015 and Developers 2021). has been used with the following parameters[4]:

- `rotation_range` - the maximum degrees of rotations in either direction
- `zoom_range` - the maximum size increase and decrease.
- `width_shift_range` - percentage of shift across the x-axis
- `height_shift_range` - percentage of shift across the y-axis
- `shear_range` - maximum skew in counterclockwise fashion in degrees
- `horizontal_flip` - allows a random horizontal flip[5]
- `fill_mode = 'nearest'` - the method of filling in the points outside the image boundaries[6]

The augmentation draws 10 samples for each image of the dataset, resulting in 7422 images. To mirror that, even though the images depicting negative cases of covid-19 were augmented, so that transformations take place over both datasets. The augmentation had proved to be a bit extreme, resulting in contorted images, and as such, was adjusted slightly, to be less extreme in all directions. Figure 4.5 depicts a sample of the more severe augmentation, whilst figure 4.6 depicts the result of the milder augmentation.

A point of concern is augmenting the images via the means of flipping (mirroring) by an axis. Horizontal flipping is completely out of the question, as that would generate nonsensical data and introduce artefacts and impact the results of any further operations. The vertical flip, however, is not as simple discarded. Human bodies are near symmetrical in general, and in that regard, flipping sounds like a sound way of augmenting the dataset. However, the heart's silhouette is captured faintly by x-rays. As such, flipping the images creates ones with faulty anatomy.

[4]The value of the parameters had been chosen per common recommendations.
[5]This will be a point of contention discussed later.
[6]Created by processes such as rotating or shifting the image.

Figure 4.5 Results of image augmentation with parameter values: `rotation_range = 30`, `zoom_range = 0.15`, `width_shift_range = 0.2`, `height_shift_range = 0.2`, `shear_range = 0.15`, `horizontal_flip = True`, `fill_mode = 'nearest'`.

Figure 4.6 Results of image augmentation with parameter values: `rotation_range = 15`, `zoom_range = 0.15`, `width_shift_range = 0.1`, `height_shift_range = 0.1`, `shear_range = 0.15`, `horizontal_flip = True`, `fill_mode = 'nearest'`.

4.3 Generation

The models are built in different frameworks, but the general idea of layer types is common across all of them. Dense layers are the most common layers. They are densely connected layers, calculating the output in this fashion: $output = activation(dot(input, kernel) + bias)$ (Chollet et al. 2015). The activation function is supplied, but is usually either a leaky rectified linear unit (leaky ReLU), a hyperbolic tangent (tanh) or a softmax[7] (usually in the output layers). The models are periodically saved to a persistent medium, evaluated and sampled.

All models were trained in different instances, with different hyperparameters, mainly a different input shape (the dimensions of the image). With some moments, changing the input shape (and by consequence, the output shape to match) required no other major change, but with some, more changes needed to be done. For instance, the DCGAN model consists of upsampling layers, and to achieve the desired output dimension, blocks of layers had to be added or removed.

Batch normalisation is another frequent layer type. As inputs, handed over to the network in batches, can have various distributions (in regards to one another within a batch), this can lead to the chase of a moving target by the network. Batch normalisation attempts to resolve this potential problem by standardising the batches, and thus stabilising the learning process. Essentially, the mean and the standard deviation of the batch is calculated, and then the batch is standardised.

Dropout layers aim to prevent overfitting models on a limited dataset by resetting (effectively ignoring) random inputs of the previous layer. On a high level, this is akin to obscuring random parts of an image during training, with the goal of learning the actual semantic features, rather than finding easy ways to conform to the training data.

4.3.1 Training parameters

[7] normalised exponential function $\sigma(\vec{z})_i = \frac{e^{z_i}}{\sum_{j=1}^{K} e^{z_j}}$, which ensures that all values sum to 1.

```
class GAN(interfaceGAN):
    def __init__(self, width, height, model_loaded=False, model_file=None):
        ...
    def train(self, epochs, batch_size, sample_interval, save_interval,
    data_path, calculate_metrics):
        ...
```

Code listing 4.5 Declaration of the training function for the GAN models.

During the training, a major setback introduced itself. The virtual machines on which the training took place often timed out, or were forcibly disconnected during the lengthy training process. At times this was due to resource exhaustion, prolonged lack of user interaction, or just general life cycle of the machines. Since some of the models take hours to train, this needed to be resolved. A parameter has been introduced, called `save_interval`, which specifies the frequency of model saving. When the model is saved to a local disk, the metrics are also calculated and logged. The reasoning behind this is a compromise between complexity and utility. The metrics are expensive to calculate at every step, and to cut that, and since the only model state that can be recalled is when the model is saved at a `save_interval`, they are calculated as the model snapshot is saved. This behaviour can be disabled, resulting in skipping of metrics calculation and saving the model only, by using the the `calculate_metrics`.

During model's instantiation the model's architecture and weights can be loaded from file by setting the `model_loaded` flag to `True` and providing a path to the model's location. Training can be then resumed from the snapshot.

Another parameter that is not to be treated as a hyperparameter, but as an indicator (or a debugging variable) is `sample_interval`, defining the interval at which images are sampled from the distribution and logged. This process is not expensive, and as such the frequency can be much higher than the `save_interval`.

The remaining parameters, `epochs, batch_size, data_path`, are straightforward, describing the number of training iterations, size of the batch in which training takes place, and the folder containing training images, respectively.

4.3.2 GAN

The vanilla GAN model's generator (figure 4.7) consists of an input layer, followed by three blocks of dense-leaky ReLu-batch normalisation layers, which in turn is connected to a dense layer. Batch normalisation is applied after every activation layer. The three hidden dense layers have 256, 512 and 1024 neurons, going from the input layer to the output layer. The activation functions (leaky ReLUs) have the parameter alpha (multiplier of negative values) set to 0.3, rather than discarding them, as is the case with a regular ReLU function. The momentum (the resistance to succumb to changes from the current batch) (Bautista 2017) is set to 0.8 in all blocks. The output layer's activation function is a hyperbolic tangent.

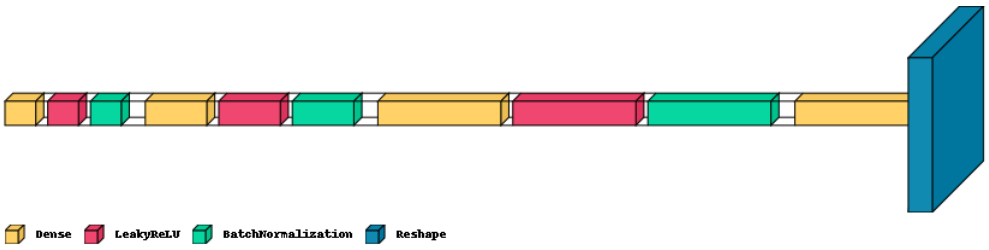

Figure 4.7 GAN generator with three blocks of layers in the hidden layer, utilising leaky ReLUs and batch normalisation.

The discriminator model (figure 4.8) is simple (which is usually the case). It consists of a flattening layer, which transforms its multidimensional input to a one dimensional array, followed by two blocks of a dense layer with a leaky ReLU activation function, with the alpha set to 0.2 in both cases. As the discriminator's output is a boolean (a binary value), stating whether the image is deemed real or synthesises, the output layer is a dense layer with a single neuron, in this case with a sigmoid activation function.

Figure 4.8 GAN discriminator with a hidden layer of two blocks and a single neuron output layer.

4.3.3 WGAN

The Wasserstein model (figure 4.9) brings convolution to the table. The generator is identical to the previous models'.

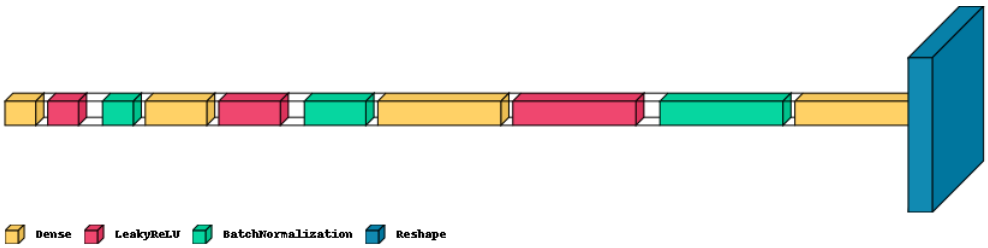

Figure 4.9 WGAN generator with two hidden layers and batch normalisation.

The critic (figure 4.10) uses convolutions instead of dense layers. Again, there are three hidden layers, with 32, 64 and 128 neurons respectively. The input layer is also a 2D convolutional layer, with 16 neurons. All convolutional layers have a kernel of size 3 and a stride size of 2. The first hidden layer also contains a zero padding layer, which adds padding of 0s around the image (this is needed to apply the convolutional filter to pixels at the edges of the image). The remaining hidden layers keep this padding. All hidden layers also include a batch normalisation with momentum = 0.8, a leaky ReLU with alpha = 0.2 and a dropout with rate = 0.25. The critic also uses a custom loss function, wasserstein loss:

```
1 import keras.backend as K
2
3 def wasserstein_loss(y_true, y_pred):
4     return K.mean(y_true * y_pred)
```

Code listing 4.6 Wasserstein loss function.

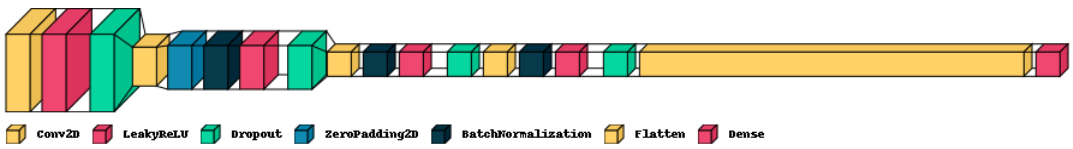

Figure 4.10 WGAN critic with three hidden convolutional layers.

The WGAN network contains 5 critics altogether, with a clip value of 0.01, as recommended in the proposing paper Arjovsky, Chintala, and Bottou 2017.

4.3.4 WGAN-GP

Though Wasserstein GAN has shown better performance in terms of convergence, quality and resistance to mode collapse, it retains some of the weaknesses, such as training instability or slow convergence (Weng 2019), caused by vanishing gradients, exploding gradients and non-continuous functions. Instead of clipping, a gradient penalty is introduced. The loss function of the critic is expanded by

$$\lambda \, \mathbb{E}_{\hat{x} \sim P_{\hat{x}}}[(\|\nabla_{\hat{x}} D(\hat{x})_2\| - 1)^2]$$

The λ parameter, called the penalty coefficient, was empirically set to 10. The original paper (Gulrajani et al. 2017) also recommends removing batch normalisation.

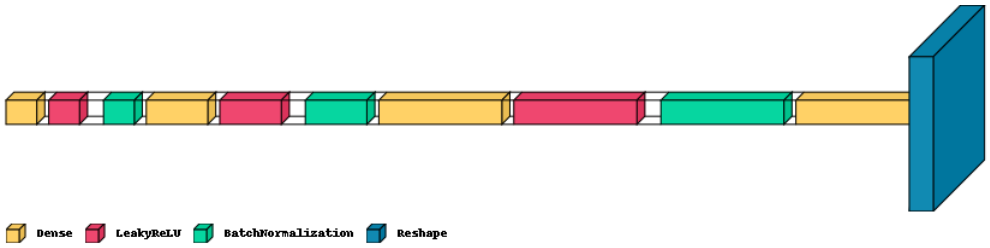

Figure 4.11 WGAN-GP generator.

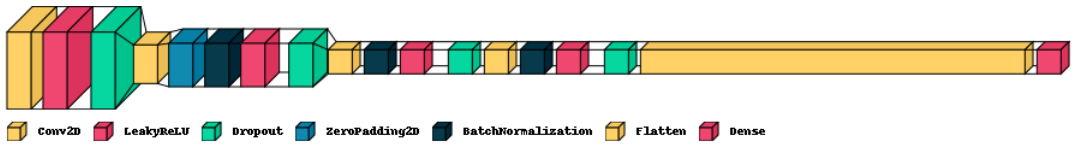

Figure 4.12 WGAN-GP critic.

4.3.5 CGAN

The premise of the conditional GAN in this regard is very similar to the concept of transfer learning. Even though the network trains on both the positive and negative labels, ultimately only the positive ones will be put to use. However, the negatively labelled images are still of the lungs, and provide more data to get the general distribution represented.

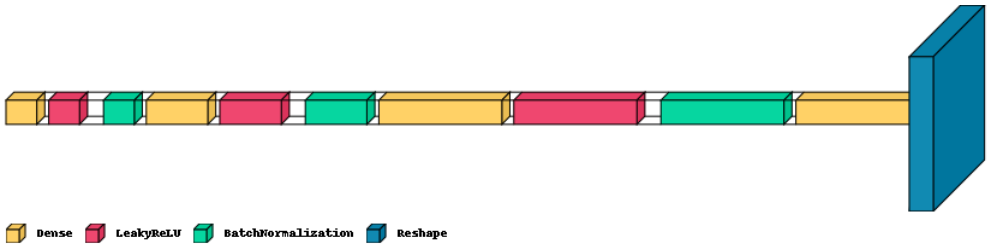

Figure 4.13 CGAN generator.

The conditional GAN's generator (figure 4.13) is based on the vanilla one's, with one difference - an input label is embedded into the input layer, with as many neurons as there are classes (in this case, 2 classes).

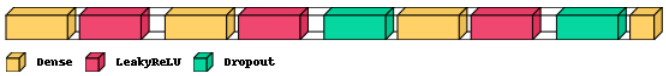

Figure 4.14 CGAN discriminator.

Similarly, the discriminator (figure 4.14) is provided with the same information in the same manner. However, instead of using batch normalisation, the discriminator contains dropout layers with a rate of 0.4 in the hidden layers.

4.3.6 DCGAN

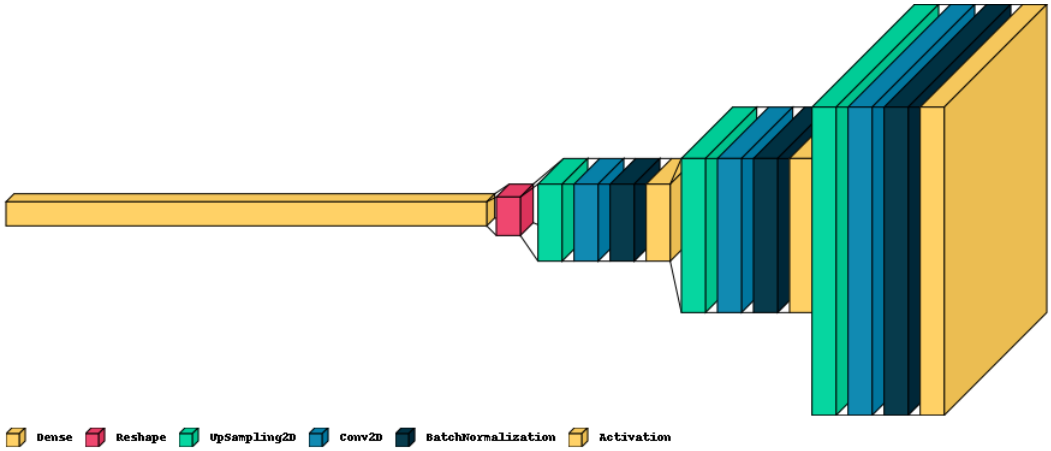

Figure 4.15 DCGAN generator with a single block.

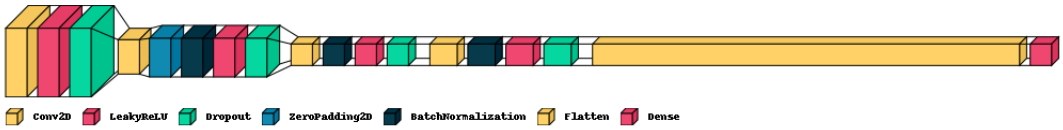

Figure 4.16 DCGAN discriminator with a single block

The deeply convoluted GAN generator (figure 4.15) consists of a number of upsampling layers followed by convolutional layers with an increasing number of neurons per layer with a 3×3 kernel. The convolutional layers' outputs are batch-wise normalised with a momentum of 0.8. The output layer's activation function is a hyperbolic tangent. Its discriminator (figure 4.16) consists of convolutional layers with a 3×3 kernel, stride of 2, a leaky ReLU activation function with alpha 0.2, batch normalisation with momentum 0.8 and a dropout rate of 0.25. The last layer has a sigmoid as the activation function.

4.3.7 VAE

To compare the performance of GANs with another generative model, a variational auto encoder has been used. The implementation of YongWook 2018 has been used as the base, but needed modifications, as it didn't function out-of-the-box. This implementation creates a custom image generator, but its image loading function `scipy.misc.imread` is deprecated, and had to be

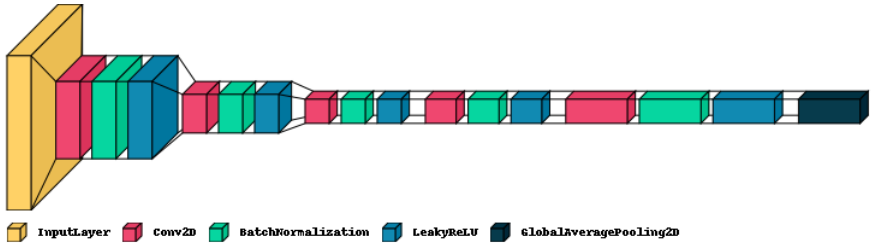

Figure 4.17 VAE encoder

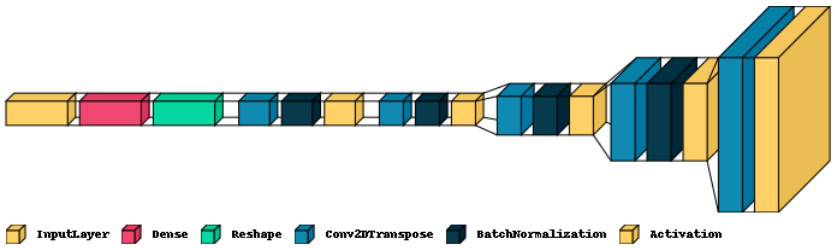

Figure 4.18 VAE decoder

replaced, by `imageio.imread` , which also enforced a conversion from a `PIL` object to a `numpy` array. Further, at each epoch's end, inception score is calculated and logged alongside the loss. Further refactoring needed to be performed, as the metrics calculation was written as a callable programme, rather than callable functions, which can be incorporated into training, which was fixed by encapsulating the code into functions and made adhered to the DRY principle. As easily visible from figures 4.17 and 4.18, it has the U-like architecture.

4.4 Classification

4.4.1 DeepCovid

The DeepCovid project (Minaee et al. 2020) trains a ResNet 18 (He et al. 2015) on a dataset with 184 images labeled as 'covid' provided by Cohen, Morrison, and Dao 2020 and 5000 non-covid images uniformly drawn from Irvin et al. 2019.

The model itself is a convolutional network of 18 layers, which is pre-trained on data from the ImageNet (Deng et al. 2009) database and provided by the PyTorch framework.

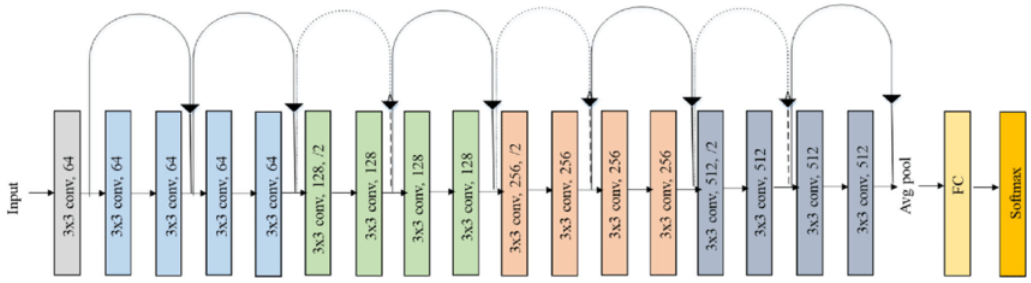

Figure 4.19 ResNet architecture, with skip connections. Source: Ramzan et al. 2019.

	train	test
positive	84	100
negative	2000	3000

Table 4.4 Class and set distribution of images used by the DeepCovid's (Minaee et al. 2020) ResNet network.

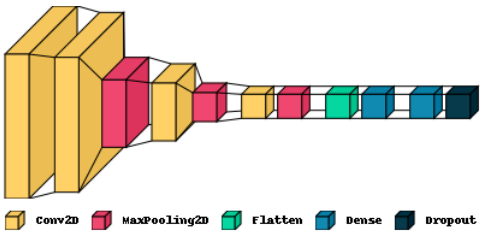

Figure 4.20 Convolutional neural network classifier. Source: Kulkarni, Walimbe, and Mundhe 2019.

4.4.2 Generic classifier network

To compare results and provide a baseline to measure against, a plain convolutional neural network classifier has also been trained. The number of layers is based on Kulkarni, Walimbe, and Mundhe 2019. The convolutional layers have, in order from start to finish, 128, 64, 32 and 32 layers, followed by a densely connected layer of 128 units and an output layer of 1 neuron with the softmax activation. The convolution filter is 3×3 for all the convolutional layers, and the pool size is 2×2.

4.5 Experiments

Logs of individual runs can be access at `https://wandb.ai/koristo1/GAN` and `https://wandb.ai/koristo1/GAN`.

4.5.1 Classifiers

The Deepcovid's ResNet-based classifier has been trained with its provided data, only split into three subsets, rather than two, to provide a universal data split for other frameworks. The model has been trained over 50 epochs, with 20 images per batch, learning rate of 0.001 and 2 workers[8].

The model is successful at identifying non-covid samples. 99.8 % non-covid images were labelled correctly. However, only 51 % of the covid positive samples were identified correctly, as visible from the confusion matrix 4.21. This can be due to the low number of samples of covid positive samples, and as such, the model has also been trained on an augmented dataset.

The false negatives are very alarming. Another training was undertaken on the same dataset, but with a different data split. Instead of the provided split, where the test set was unreasonably large (3100 images), the images have been pooled and then randomly, without replacement, 60 % of the images were designated as training data, 20 % as validation and the remaining 20 % as test data. This process was done over labels separately, so the proportions of covid and non-covid is the same. This has resulted in a much lower number of false negatives and no false negatives, as visible in the confusion matrix 4.22.

Hoping for a fair comparison with a naive model, a simple convolutional network was also trained on the same dataset. However, the model has always quickly degenerated and collapsed,

[8]The number of subprocesses for data loading.

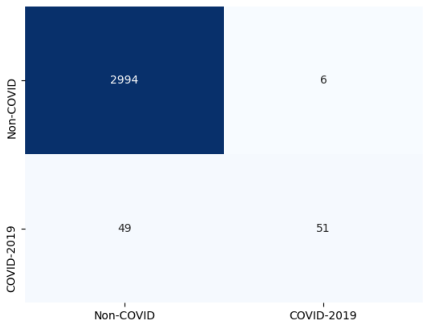

Figure 4.21 Confusion matrix of ResNet-18 with the provided test set.

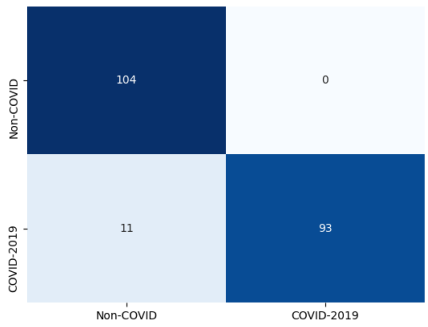

Figure 4.22 Confusion matrix of ResNet-18 on a balanced test set.

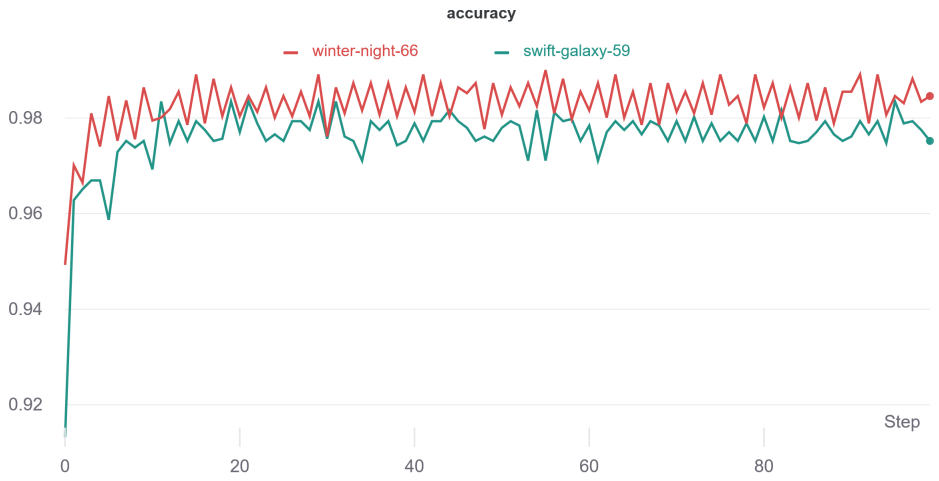

Figure 4.23 Accuracy of the Deep covid models during their training. The red line tracks the training on a larger training dataset, whilst the green line tracks training on the original dataset.

resulting in 0 loss, and hence the loss function, which uses the logarithmic value of that, escalated into infinity. Thus, no matter what the activation functions were, this classifier is unusable - both the validation and test results were 50 %.

Initially, all GAN models were trained with a batch size of 32, 10000 epochs and latent dimension of 100. The desired output (and by extension, required input) was i.e. greyscale images. Models were trained on both the regular dataset, sourced from Wang, Lin, and Wong 2020, as well as the augmented dataset[9].

There is a dramatic difference in training time based on the model and desired image dimensions. Further, the captured duration of training does not include data loading, only the training of the network itself. Due to network transfer from a cloud storage to a virtual machine's, the loading process can take up to an hour, if not longer, depending on the dataset size. This in general hinders easy reproducibility and tuning, and could be solved by compressing the individual sets into archives and transferring those instead to dramatically reduce copying by removing a large chunk of the copying overhead.

[9] As described in section 4.2.3.

Model type	Run time (s)	Batch size	Epochs	Height	Width	Dataset
wgan-gp	5913	32	10000.0	64.0	64.0	normal
dcgan	8967	32	10000.0	256.0	256.0	normal
wgan	9042	32	10000.0	256.0	256.0	normal
cgan	818	32	10000.0	256.0	256.0	normal
wgan	2687	32	10000.0	64.0	64.0	normal
cgan	2221	32	10000.0	64.0	64.0	normal
wgan-gp	6000	32	10000.0	256.0	256.0	normal
wgan-gp	3029	32	10000.0	64.0	64.0	augmented
wgan	3394	32	10000.0	64.0	64.0	augmented
cgan	4498	32	10000.0	64.0	64.0	augmented
wgan-gp	6285	32	10000.0	256.0	256.0	augmented
wgan	10655	32	10000.0	256.0	256.0	augmented
cgan	4751	32	10000.0	256.0	256.0	augmented

Table 4.5 Fixed dataset example

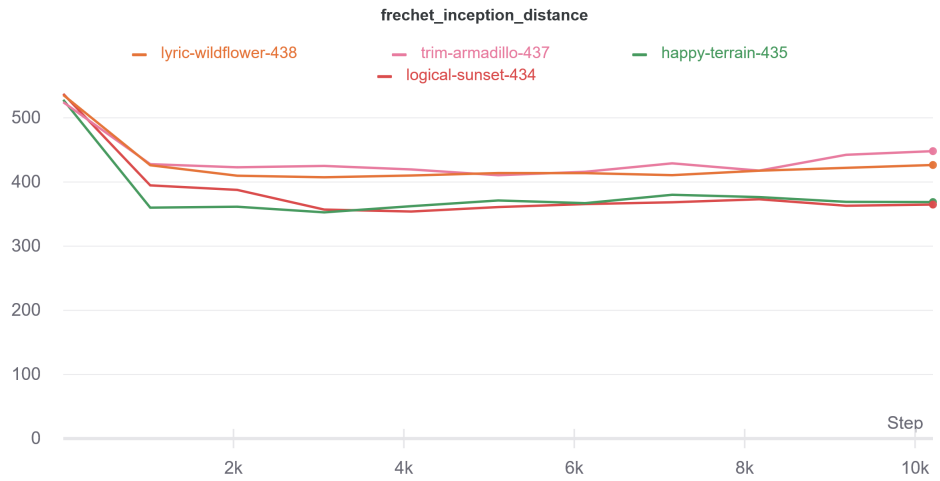

Figure 4.24 FID of GAN models. The orange and pink curves are of models trained on the augmented dataset, whilst the green and red are trained on the regular dataset. Further, the orange and red curves are trained to generate images with dimensions 64×64, and the green and pink with dimensions 256×256.

4.5.2 GAN

The vanilla GAN has tangible results, especially on the non-augmented dataset (figures 4.27 and 4.28).

As visible in figure 4.24, all the GAN models have a similar curve when it comes to Fréchet inception distance, having a steep decrease in the first 1000 epochs and then converging from 2000th epoch onwards[10].

4.5.3 CGAN

The conditional dataset performs quite well on the regular dataset, creating tangible pictures at the higher resolution (4.48), and slightly worse, but still recognizable, at lower (4.45). This is due to the much larger dataset size, which also learns from negative samples, which in general are

[10]The lower the Fréchet inception distance, the better.

Figure 4.25 GAN results after 10000 epochs on the augmented dataset with images of size 64×64.

Figure 4.26 GAN results after 10000 epochs on the augmented dataset with images of size 256×256.

Figure 4.27 GAN results after 10000 epochs on the normal dataset with images of size 64×64.

Figure 4.28 GAN results after 10000 epochs on the normal dataset with images of size 256×256.

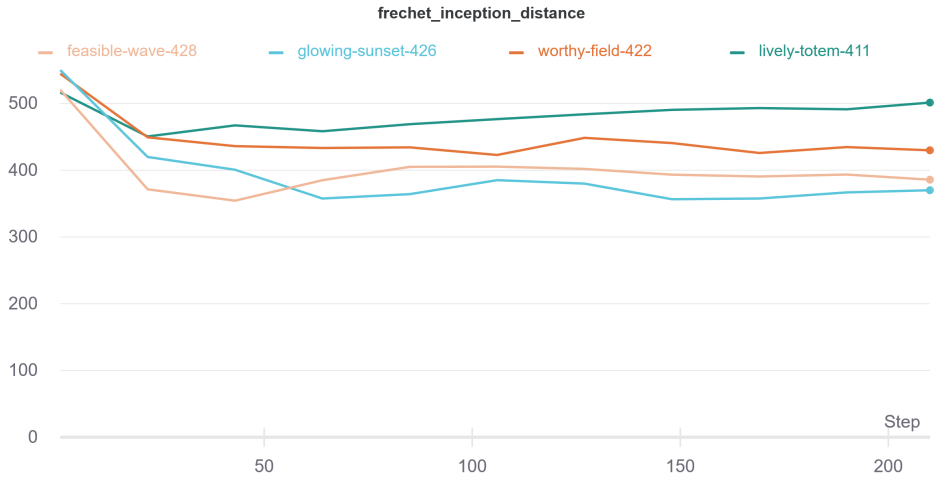

Figure 4.29 FID of conditional GAN models. The cyan and orange models produce 64×64images, the first of which was trained on the regular dataset, and the latter on the augmented dataset. The complementing datasets generate images with dimensions 256×256, where the peach and brown-grey curves correspond to the original dataset and the augmented dataset, respectively.

quite similar (in terms of the human physiology). The augmented dataset's performance with a conditional GAN, whist comparing worse with the regular dataset, seems more promising than the WGAN.

As was the case with the vanilla GAN models, the curves of FID throughout the training are all quite similar, dipping steeply early and then having a very mild rise.

4.5.4 WGAN

Wasserstein GAN has unfortunately fallen short in general, especially with gradient penalty. Though the 'regular' WGAN generates results that with a grain of salt resemble the domain they are trained on (figures 4.34,4.35,4.32), it also easily generates irrelevant data (figure 4.33). Even worse, the WGAN with gradient penalty performs even worse, generating only and only white noise (4.36,4.33,4.34 and 4.35). These models were given a second chance to perform with fewer epochs, to see if they produce better results earlier during the training process.

FID of both families (with and without gradient penalty loss) show wild oscillation during training, moreso in the case of models with gradient penalty. Whilst they do have the steepest initial drop in value, as visible in figure 4.31, they follow by a quick rise and unstable progress. The regular WGAN models perform slightly better, but still not adequately, as depicted in figure 4.30, the values do not unfold in a controlled manner.

Based on the recorded metrics, the 2500th epoch was determined most suitable for a second examination.

4.5.5 DCGAN

Similarly to the Wasserstein GANs, the DCGAN is a failure, and on top of that it has suffered from a mode collapse. This can be easily visible in figures 4.43, 4.41 and 4.44, where the generated images are identical or near identical. The expected result of the DCGAN was abnormal and distorted body shapes depicted in the image, as reported by Kovalev and Kazlouski 2019.

Regarding FID, the models quickly converge, most likely due to the mode collapse visible in the result images. The run that had the lowest FID has basically not changed from the 2000th

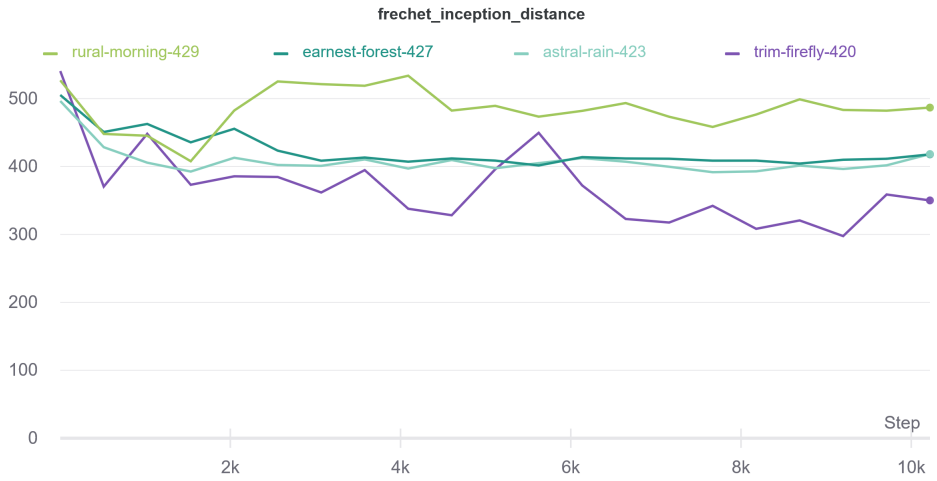

Figure 4.30 FID of WGAN models. The lime and brown-grey models represent training runs on the normal dataset, with image dimensions of $256 \times 256$and 64×64, respetively. The other two, purple and green, represent runs on the augmented dataset, again, with dimensions $256 \times 256$and 64×64.

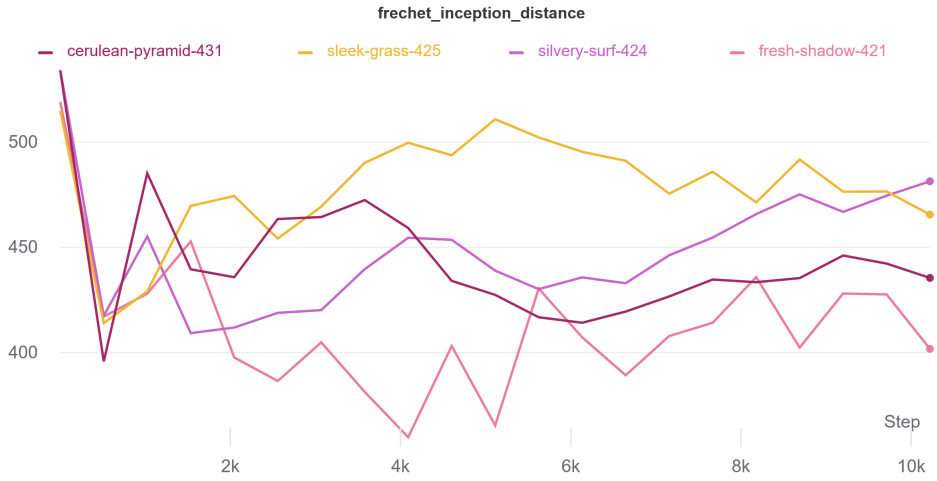

Figure 4.31 FID of WGAN models with a gradient penalty loss. The crimson and yellow curves depict the FID during training on the normal dataset with image dimensions $64 \times 64$and 256×256successively, magenta and pink follow the same dimensions on the augmented dataset.

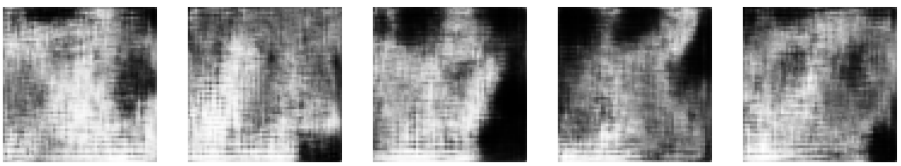

Figure 4.32 Wasserstein GAN results after 10000 epochs on the augmented dataset with images of size 64×64.

Figure 4.33 Wasserstein GAN results after 10000 epochs on the augmented dataset with images of size 256×256.

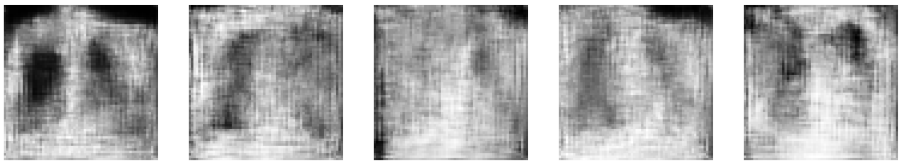

Figure 4.34 Wasserstein GAN results after 10000 epochs on the normal dataset with images of size 64×64.

Figure 4.35 Wasserstein GAN results after 10000 epochs on the normal dataset with images of size 256×256.

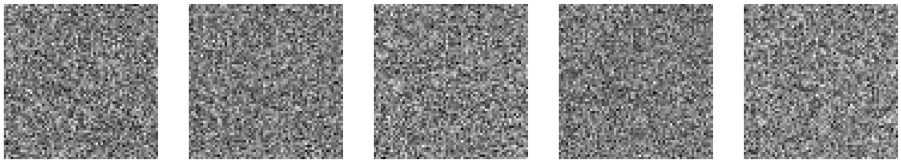

Figure 4.36 Wasserstein GAN with gradient penalty results after 10000 epochs on the augmented dataset with images of size 64×64.

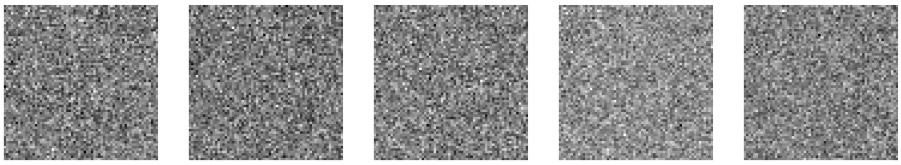

Figure 4.37 Wasserstein GAN with gradient penalty results after 10000 epochs on the augmented dataset with images of size 256×256.

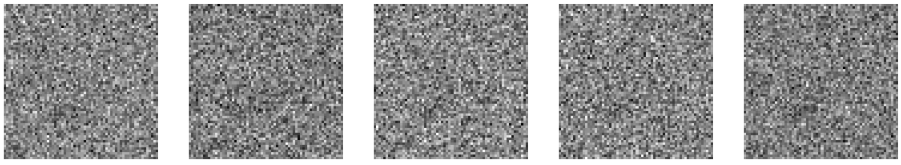

Figure 4.38 Wasserstein GAN with gradient penalty results after 10000 epochs on the normal dataset with images of size 64×64.

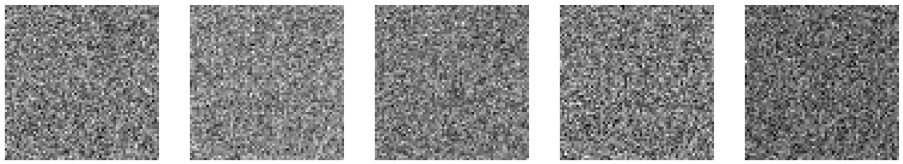

Figure 4.39 DCGAN results after 10000 epochs on the normal dataset with images of size 256×256.

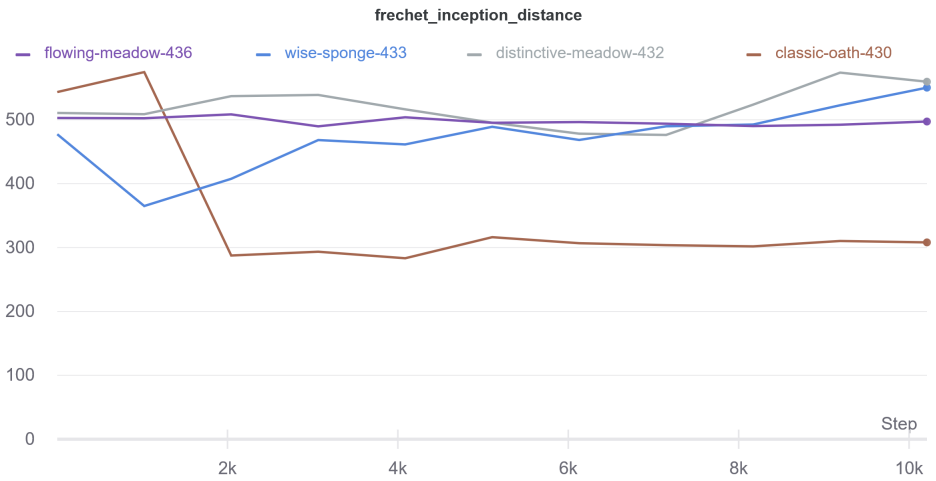

Figure 4.40 FID of DCGAN models. The quick convergence is alarming. The purple and blue lines track training with $256 \times 256$and 64×64, respectively, on the augmented dataset. The gray and brown lines then track training on the regular dataset, with $64 \times 64$and 256×256.

epoch onwards, whilst the others increased in value over time. In the same vein, the variational auto encoder performed terribly, producing but noise and no usable data, and was not taken into consideration going gorward.

4.5.6 Second wind

As mentioned during the examination of models' results, some of them have failed to model the desired domain. That could be an issue of mode collapse, overfitting or simply general lack of data. Attempting to salvage, the models were loaded at an earlier checkpoint, where their metrics were promising, and examined. However, this has not come into fruition, as the results are still a scrambled mess, as visible in figure 4.49.

Through and through, the vanilla GAN model, trained on the base dataset with image dimensions of 256×256has emerged superior amongst its peers. The bets performing dataset of DeepCovid contained but 312 image, and so a combination of those 312 and 312 generated images was used to train the DeepCovid model.

Figure 4.41 DCGAN results after 10000 epochs on the augmented dataset with images of size 64×64.

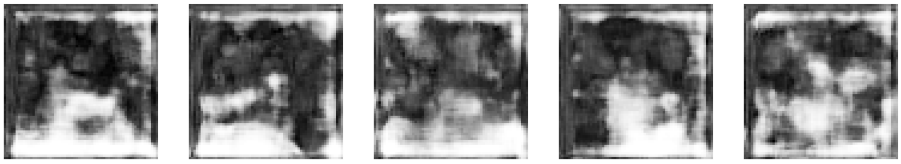

Figure 4.42 DCGAN results after 10000 epochs on the augmented dataset with images of size 256 × 256.

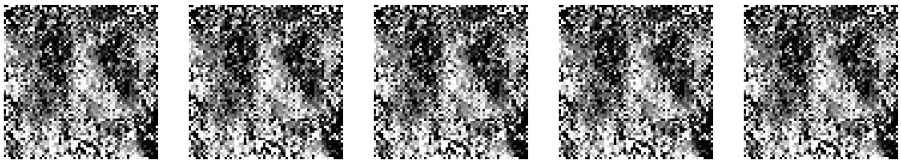

Figure 4.43 DCGAN results after 10000 epochs on the normal dataset with images of size 64 × 64.

Figure 4.44 Wasserstein GAN with gradient penalty results after 10000 epochs on the normal dataset with images of size 256 × 256.

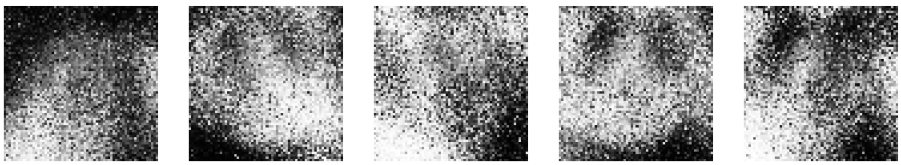

Figure 4.45 Conditional GAN results after 10000 epochs on the augmented dataset with images of size 64 × 64.

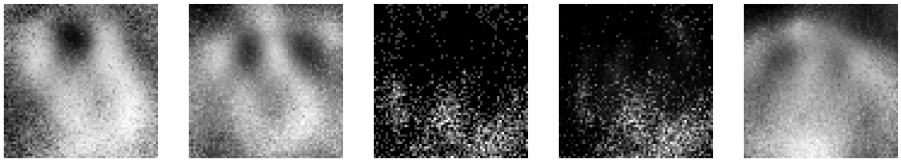

Figure 4.46 Conditional GAN results after 10000 epochs on the augmented dataset with images of size 256 × 256.

Figure 4.47 Conditional GAN results after 10000 epochs on the regular dataset with images of size 64 × 64.

Figure 4.48 Conditional GAN results after 10000 epochs on the normal dataset with images of size 256×256.

Figure 4.49 Images generated at earlier stages of the training. From left to right, they are as following: Wasserstein GAN with gradient penalty at 3000th epoch, Deep Convoluted GAN at 2000th epoch and Wasserstein GAN at 4500th epoch.

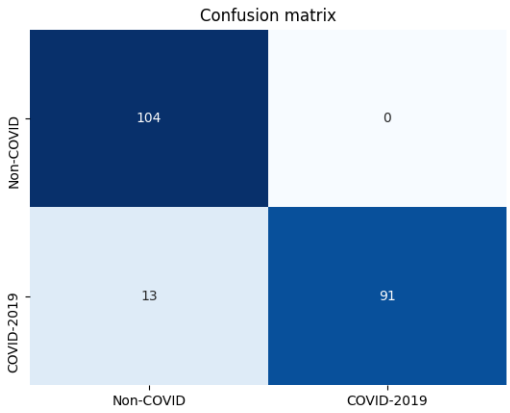

Figure 4.50 Confusion matrix of the DeepCovid classifier with the original covid data supplemented by data generated by a GAN model.

Chapter 5

Conclusion

The research aimed to review state-of-the-art techniques used in the domain of medical imaging, with a focus on the application of generative models and segmentation tasks. As proven times and times again in their short life span, adversarial models are a remarkable tool for exploring the intricate distributions of complex data, and have established a foothold in medical imaging, competing with modern approaches.

The experiments pick a handful of the most common models for implementation and evaluation, to supplement the dissatisfactory amount of positive pathologies of covid-19, and those models were compared and contrasted against each other. The models were ran and evaluated with regular dataset, as well as their augmented variations. The expectations were such that the augmented datasets will bolster the models, by tackling overfitting and forcing the models to learn the semantic information from images, but in the end, the baseline dataset outperformed the augmented in every sense. Similarly, the more refined models performed much worse than their simpler brethren.

The generated images from these models have demonstrated being on par with real data when supplementing it during training of classification neural networks. Using data generated from the best performing model - the vanilla GAN model without any additional loss functions or modifications, on the non-augmented dataset, working with greyscale 256×256 images to train an existing classifier ResNet-18, the model sports a 93% accuracy.

Further tinkering with model shapes and hyperparameters and an increase in volume of training data shows promises of further improvement and practical use of synthetic data.

Appendix A

GAN evaluation metrics

Compiled by Borji 2018.

1. Average Log-likelihood
2. Coverage Metric
3. Inception Score (IS)
4. Modified Inception Score (m-IS)
5. Mode Score
6. AM Score
7. Frechet Inception Distance (FID)
8. Maximum Mean Discrepancy (MMD)
9. The Wasserstein Critic
10. Birthday Paradox Test
11. Classifier Two-sample Tests (C2ST)
12. Classification Performance
13. Boundary Distortion
14. Number of Statistically-Different Bins (NDB)
15. Image Retrieval Performance
16. Generative Adversarial Metric (GAM)
17. Tournament Win Rate and Skill Rating
18. Normalized Relative Discriminative Score (NRDS)
19. Adversarial Accuracy and Adversarial Divergence
20. Geometry Score
21. Reconstruction Error

22. Image Quality Measures (SSIM, PSNR and Sharpness Difference)
23. Low-level Image Statistics
24. Precision, Recall and F1 Score

Appendix B

CovidNet datasets

- https://github.com/ieee8023/covid-chestxray-dataset.git
- https://github.com/agchung/Figure1-COVID-chestxray-dataset.git
- https://github.com/agchung/Actualmed-COVID-chestxray-dataset.git
- https://www.kaggle.com/tawsifurrahman/covid19-radiography-database/discussion/209607 (broken)
- https://drive.google.com/file/d/1xt7g5LkZuX09e1a8rK9sRXIrGFN6rjzl/view?usp=sharing (functional alternative)
- https://www.kaggle.com/c/rsna-pneumonia-detection-challenge/data

Bibliography

[1] David Abramian and Anders Eklund. "Refacing: Reconstructing Anonymized Facial Features Using GANS". In: *2019 IEEE 16th International Symposium on Biomedical Imaging (ISBI 2019)* (Apr. 2019). DOI: 10.1109/isbi.2019.8759515. URL: http://dx.doi.org/10.1109/ISBI.2019.8759515.

[2] Jafar Alzubi, Anand Nayyar, and Akshi Kumar. "Machine learning from theory to algorithms: an overview". In: *Journal of physics: conference series*. Vol. 1142. 1. IOP Publishing. 2018, p. 012012.

[3] Pujitha Appan K. and Jayanthi Sivaswamy. "Retinal Image Synthesis for CAD Development". In: *Image Analysis and Recognition*. Ed. by lio Campilho Auré, Fakhri Karray, and Bart ter Haar Romeny. Cham: Springer International Publishing, 2018, pp. 613–621. ISBN: 978-3-319-93000-8.

[4] Martin Arjovsky, Soumith Chintala, and Léon Bottou. *Wasserstein GAN*. 2017. arXiv: 1701.07875 [stat.ML].

[5] Karim Armanious et al. "MedGAN: Medical image translation using GANs". In: *Computerized Medical Imaging and Graphics* 79 (Jan. 2020), p. 101684. ISSN: 0895-6111. DOI: 10.1016/j.compmedimag.2019.101684. URL: http://dx.doi.org/10.1016/j.compmedimag.2019.101684.

[6] Valérie Aumage. *Health data and data privacy. Challenges for data processors under the GDPR*. June 2016. URL: https://globaldatahub.taylorwessing.com/article/health-data-and-data-privacy-challenges-for-data-processors-under-the-gdpr.

[7] Shane Barratt and Rishi Sharma. *A Note on the Inception Score*. 2018. arXiv: 1801.01973 [stat.ML].

[8] Christoph Baur, Shadi Albarqouni, and Nassir Navab. *Generating Highly Realistic Images of Skin Lesions with GANs*. 2018. arXiv: 1809.01410 [cs.CV].

[9] Christoph Baur, Shadi Albarqouni, and Nassir Navab. *MelanoGANs: High Resolution Skin Lesion Synthesis with GANs*. 2018. arXiv: 1804.04338 [cs.CV].

[10] Joseph J. Bautista. *A Bit Beyond Gradient Descent: Mini-Batch, Momentum, and Some Dude Named Yuri Nesterov*. Dec. 2017. URL: https://towardsdatascience.com/a-bit-beyond-gradient-descent-mini-batch-momentum-and-some-dude-named-yuri-nesterov-a3640f9e496b.

[11] Andrew Beers et al. *High-resolution medical image synthesis using progressively grown generative adversarial networks*. 2018. arXiv: 1805.03144 [cs.CV].

[12] Daniel J Bell. 2020. URL: https://radiopaedia.org/articles/radiological-image-artifact.

[13] Avi Ben-Cohen et al. "Virtual PET Images from CT Data Using Deep Convolutional Networks: Initial Results". In: *Lecture Notes in Computer Science* (2017), pp. 49–57. ISSN: 1611-3349. DOI: 10.1007/978-3-319-68127-6_6. URL: http://dx.doi.org/10.1007/978-3-319-68127-6_6.

[14] Abi Berger. "Positron emission tomography". In: *BMJ* 326.7404 (June 2003), pp. 1449–1449. DOI: 10.1136/bmj.326.7404.1449. URL: https://doi.org/10.1136/bmj.326.7404.1449.

[15] Lei Bi et al. *Synthesis of Positron Emission Tomography (PET) Images via Multi-channel Generative Adversarial Networks (GANs)*. 2017. arXiv: 1707.09747 [cs.CV].

[16] Lukas Biewald. *Experiment Tracking with Weights and Biases*. Software available from wandb.com. 2020. URL: https://www.wandb.com/.

[17] Ali Borji. *Pros and Cons of GAN Evaluation Measures*. 2018. arXiv: 1802.03446 [cs.CV].

[18] Tomas Borovicka. *Multi-agent systems and The Game Theory. Games in Normal Form, Games in Extensive Form*. https://courses.fit.cvut.cz/BI-ZUM/media/en/lectures/10-games-noanim.pdf. Thákurova 9, 160 00 Prague 6: Department of Theoretical Computer Science , Faculty of Information Technology Czech Technical University in Prague, 2013.

[19] Christopher Bowles et al. *GAN Augmentation: Augmenting Training Data using Generative Adversarial Networks*. 2018. arXiv: 1810.10863 [cs.CV].

[20] Francesco Calimeri et al. "Biomedical Data Augmentation Using Generative Adversarial Neural Networks". In: *Artificial Neural Networks and Machine Learning – ICANN 2017*. Ed. by Alessandra Lintas et al. Cham: Springer International Publishing, 2017, pp. 626–634. ISBN: 978-3-319-68612-7.

[21] François Chollet et al. *Keras*. https://keras.io. 2015.

[22] Maria J. M. Chuquicusma et al. *How to Fool Radiologists with Generative Adversarial Networks? A Visual Turing Test for Lung Cancer Diagnosis*. 2018. arXiv: 1710.09762 [cs.CV].

[23] Özgün Çiçek et al. *3D U-Net: Learning Dense Volumetric Segmentation from Sparse Annotation*. 2016. arXiv: 1606.06650 [cs.CV].

[24] Joseph Paul Cohen, Paul Morrison, and Lan Dao. "COVID-19 image data collection". In: *arXiv 2003.11597* (2020). URL: https://github.com/ieee8023/covid-chestxray-dataset.

[25] Pedro Costa et al. "End-to-End Adversarial Retinal Image Synthesis". In: *IEEE Transactions on Medical Imaging* 37.3 (2018), pp. 781–791. DOI: 10.1109/TMI.2017.2759102.

[26] Wei Dai et al. *SCAN: Structure Correcting Adversarial Network for Organ Segmentation in Chest X-rays*. 2017. arXiv: 1703.08770 [cs.CV].

[27] Shibsankar Das. *Image similarity using Triplet Loss*. July 2019. URL: https://towardsdatascience.com/image-similarity-using-triplet-loss-3744c0f67973.

[28] Jia Deng et al. "Imagenet: A large-scale hierarchical image database". In: *2009 IEEE conference on computer vision and pattern recognition*. Ieee. 2009, pp. 248–255. URL: http://www.image-net.org/.

[29] TensorFlow Developers. "TensorFlow". In: (Mar. 2021). DOI: 10.5281/zenodo.4758419.

[30] Qi Dou et al. *3D Deeply Supervised Network for Automatic Liver Segmentation from CT Volumes*. 2016. arXiv: 1607.00582 [cs.CV].

[31] Vincent Dumoulin and Francesco Visin. "A guide to convolution arithmetic for deep learning". In: *ArXiv e-prints* (Mar. 2016). eprint: 1603.07285.

[32] IBM Cloud Education. *Unsupervised learning*. Sept. 2020. URL: https://www.ibm.com/cloud/learn/unsupervised-learning.

[33] Alon Efrat et al. "New Similarity Measures between Polylines with Applications to Morphing and Polygon Sweeping". In: *Discrete and Computational Geometry* 28 (July 2002), pp. 535–569. DOI: 10.1007/s00454-002-2886-1.

[34] Hajar Emami et al. "Generating synthetic CTs from magnetic resonance images using generative adversarial networks". In: *Medical Physics* 45.8 (2018), pp. 3627–3636. DOI: https://doi.org/10.1002/mp.13047. eprint: https://aapm.onlinelibrary.wiley.com/doi/pdf/10.1002/mp.13047. URL: https://aapm.onlinelibrary.wiley.com/doi/abs/10.1002/mp.13047.

[35] Farzan Farnia and Asuman Ozdaglar. *GANs May Have No Nash Equilibria*. 2020. arXiv: 2002.09124 [cs.LG].

[36] *Gamma Knife*. Jan. 2018. URL: https://www.radiologyinfo.org/en/info/gamma_knife.

[37] Ian J. Goodfellow et al. *Generative Adversarial Networks*. 2014. arXiv: 1406.2661 [stat.ML].

[38] John T. Guibas, Tejpal S. Virdi, and Peter S. Li. *Synthetic Medical Images from Dual Generative Adversarial Networks*. 2018. arXiv: 1709.01872 [cs.CV].

[39] Ishaan Gulrajani et al. *Improved Training of Wasserstein GANs*. 2017. arXiv: 1704.00028 [cs.LG].

[40] Changhee Han et al. "GAN-based synthetic brain MR image generation". In: *2018 IEEE 15th International Symposium on Biomedical Imaging (ISBI 2018)*. 2018, pp. 734–738. DOI: 10.1109/ISBI.2018.8363678.

[41] Kaiming He et al. *Deep Residual Learning for Image Recognition*. 2015. arXiv: 1512.03385 [cs.CV].

[42] Yuta Hiasa et al. "Cross-Modality Image Synthesis from Unpaired Data Using CycleGAN". In: *Simulation and Synthesis in Medical Imaging*. Ed. by Ali Gooya et al. Cham: Springer International Publishing, 2018, pp. 31–41. ISBN: 978-3-030-00536-8.

[43] Avinash Hindupur. *GAN Zoo*. https://github.com/hindupuravinash/the-gan-zoo. 2017.

[44] *Imaging and radiology*. 1997. URL: https://medlineplus.gov/ency/article/007451.htm.

[45] Talha Iqbal and Hazrat Ali. "Generative Adversarial Network for Medical Images (MI-GAN)". In: *Journal of Medical Systems* 42.11 (Oct. 2018). ISSN: 1573-689X. DOI: 10.1007/s10916-018-1072-9. URL: http://dx.doi.org/10.1007/s10916-018-1072-9.

[46] Jeremy Irvin et al. *CheXpert: A Large Chest Radiograph Dataset with Uncertainty Labels and Expert Comparison*. 2019. arXiv: 1901.07031 [cs.CV].

[47] Phillip Isola et al. *Image-to-Image Translation with Conditional Adversarial Networks*. 2018. arXiv: 1611.07004 [cs.CV].

[48] Yangqing Jia et al. "Caffe: Convolutional Architecture for Fast Feature Embedding". In: *arXiv preprint arXiv:1408.5093* (2014).

[49] Cheng-Bin Jin et al. "Deep CT to MR Synthesis Using Paired and Unpaired Data". In: *Sensors* 19.10 (2019). ISSN: 1424-8220. DOI: 10.3390/s19102361. URL: https://www.mdpi.com/1424-8220/19/10/2361.

[50] James M. Joyce. "Kullback-Leibler Divergence". In: *International Encyclopedia of Statistical Science*. Ed. by Miodrag Lovric. Berlin, Heidelberg: Springer Berlin Heidelberg, 2011, pp. 720–722. ISBN: 978-3-642-04898-2. DOI: 10.1007/978-3-642-04898-2_327. URL: https://doi.org/10.1007/978-3-642-04898-2_327.

[51] Horst-Michael Gross Klaus Debes Alexander Koenig. "Transfer Functions in Artificial Neural Networks - A Simulation-Based Tutorial". In: *Brains, Minds and Media* 1 (2005).

[52] Dimitrios Korkinof et al. *High-Resolution Mammogram Synthesis using Progressive Generative Adversarial Networks*. 2019. arXiv: 1807.03401 [cs.CV].

[53] Vassili Kovalev and Siarhei Kazlouski. *Examining the Capability of GANs to Replace Real Biomedical Images in Classification Models Training*. 2019. arXiv: 1904.08688 [cs.CV].

[54] Sumedh Kulkarni, Pranav Walimbe, and Prathamesh Mundhe. *IE 7615: NeuralNetworks and Deep Learning - Final Project*. https://github.com/sumedhkulkarni7/Image-Classification-using-CNN-Keras-and-Tensorflow-in-Python. 2019.

[55] Avisek Lahiri et al. *Retinal Vessel Segmentation under Extreme Low Annotation: A Generative Adversarial Network Approach*. 2018. arXiv: 1809.01348 [cs.CV].

[56] Zhengchun Liu et al. "TomoGAN: low-dose synchrotron x-ray tomography with generative adversarial networks: discussion". In: *Journal of the Optical Society of America A* 37.3 (Feb. 2020), p. 422. ISSN: 1520-8532. DOI: 10.1364/josaa.375595. URL: http://dx.doi.org/10.1364/JOSAA.375595.

[57] Andrew L Maas, Awni Y Hannun, and Andrew Y Ng. "Rectifier nonlinearities improve neural network acoustic models". In: *Proc. icml*. Vol. 30. 1. Citeseer. 2013, p. 3.

[58] Ali Madani et al. "Semi-supervised learning with generative adversarial networks for chest X-ray classification with ability of data domain adaptation". In: *2018 IEEE 15th International Symposium on Biomedical Imaging (ISBI 2018)*. 2018, pp. 1038–1042. DOI: 10.1109/ISBI.2018.8363749.

[59] Dwarikanath Mahapatra and Behzad Bozorgtabar. *Progressive Generative Adversarial Networks for Medical Image Super resolution*. 2019. arXiv: 1902.02144 [cs.CV].

[60] Dwarikanath Mahapatra, Behzad Bozorgtabar, et al. *Efficient Active Learning for Image Classification and Segmentation using a Sample Selection and Conditional Generative Adversarial Network*. 2019. arXiv: 1806.05473 [cs.CV].

[61] Faisal Mahmood, Richard Chen, and Nicholas J. Durr. "Unsupervised Reverse Domain Adaptation for Synthetic Medical Images via Adversarial Training". In: *IEEE Transactions on Medical Imaging* 37.12 (Dec. 2018), pp. 2572–2581. ISSN: 1558-254X. DOI: 10.1109/tmi.2018.2842767. URL: http://dx.doi.org/10.1109/TMI.2018.2842767.

[62] Matteo Maspero et al. "Dose evaluation of fast synthetic-CT generation using a generative adversarial network for general pelvis MR-only radiotherapy". In: *Physics in Medicine & Biology* 63.18 (Sept. 2018), p. 185001. ISSN: 1361-6560. DOI: 10.1088/1361-6560/aada6d. URL: http://dx.doi.org/10.1088/1361-6560/aada6d.

[63] Hugo Mayo et al. *Types of Medical Imaging*. 2018. URL: https://www.doc.ic.ac.uk/~jce317/types-medical-imaging.html.

[64] The National Institute of Mental Health. "Computed Tomography (CT)". In: (July 2013). URL: https://www.nibib.nih.gov/science-education/science-topics/computed-tomography-ct.

[65] The National Institute of Mental Health. "Ultrasound Imaging". In: (July 2016). URL: https://www.nibib.nih.gov/science-education/science-topics/ultrasound.

[66] Merriam-Webster. URL: https://www.merriam-webster.com/dictionary/in%5C%20vivo (visited on 03/12/2021).

[67] Ethan Bueno de Mesquita. *Political Economy for Public Policy*. Princeton University Press, 2016. ISBN: 9780691168739. DOI: 10.2307/j.ctvc772fr. URL: http://www.jstor.org/stable/j.ctvc772fr.

[68] Fausto Milletari, Nassir Navab, and Seyed-Ahmad Ahmadi. *V-Net: Fully Convolutional Neural Networks for Volumetric Medical Image Segmentation*. 2016. arXiv: 1606.04797 [cs.CV].

[69] Shervin Minaee et al. "Deep-COVID: Predicting COVID-19 From Chest X-Ray Images Using Deep Transfer Learning". In: *arXiv preprint arXiv:2004.09363* (2020).

[70] Mehdi Mirza and Simon Osindero. *Conditional Generative Adversarial Nets*. 2014. arXiv: 1411.1784 [cs.LG].

[71] Tony C. W. Mok and Albert C. S. Chung. "Learning Data Augmentation for Brain Tumor Segmentation with Coarse-to-Fine Generative Adversarial Networks". In: *Lecture Notes in Computer Science* (2019), pp. 70–80. ISSN: 1611-3349. DOI: 10.1007/978-3-030-11723-8_7. URL: http://dx.doi.org/10.1007/978-3-030-11723-8_7.

[72] *MRI scan*. Aug. 2018. URL: https://www.nhs.uk/conditions/mri-scan/.

[73] Dong Nie, Roger Trullo, et al. *Medical Image Synthesis with Context-Aware Generative Adversarial Networks*. 2016. arXiv: 1612.05362 [cs.CV].

[74] Dong Nie, Lei Xiang, et al. *Dual Adversarial Learning with Attention Mechanism for Fine-grained Medical Image Synthesis*. 2019. arXiv: 1907.03297 [eess.IV].

[75] Adam Paszke et al. "PyTorch: An Imperative Style, High-Performance Deep Learning Library". In: *Advances in Neural Information Processing Systems 32*. Curran Associates, Inc., 2019, pp. 8024–8035. URL: http://papers.neurips.cc/paper/9015-pytorch-an-imperative-style-high-performance-deep-learning-library.pdf.

[76] Andrew J. Plassard et al. "Learning implicit brain MRI manifolds with deep learning". In: *Medical Imaging 2018: Image Processing* (Mar. 2018). Ed. by Elsa D. Angelini and Bennett A.Editors Landman. DOI: 10.1117/12.2293515. URL: http://dx.doi.org/10.1117/12.2293515.

[77] *Positron Emission Tomography - Computed Tomography (PET/CT)*. Aug. 2019. URL: https://www.radiologyinfo.org/en/info/pet.

[78] David C Preston. "Magnetic resonance imaging (mri) of the brain and spine: Basics". In: *MRI Basics, Case Med* 30 (2006).

[79] Alec Radford, Luke Metz, and Soumith Chintala. 2016. arXiv: 1511.06434 [cs.LG].

[80] Farheen Ramzan et al. "A Deep Learning Approach for Automated Diagnosis and Multi-Class Classification of Alzheimer's Disease Stages Using Resting-State fMRI and Residual Neural Networks". In: *Journal of Medical Systems* 44.2 (Dec. 2019), p. 37. ISSN: 1573-689X. DOI: 10.1007/s10916-019-1475-2. URL: https://doi.org/10.1007/s10916-019-1475-2.

[81] Olaf Ronneberger, Philipp Fischer, and Thomas Brox. *U-Net: Convolutional Networks for Biomedical Image Segmentation*. 2015. arXiv: 1505.04597 [cs.CV].

[82] Stuart J. Russell and Peter Norvig. *Artificial Intelligence: A Modern Approach (2nd Edition)*. Prentice Hall, Dec. 2002. ISBN: 0137903952.

[83] Carl F. Sabottke and Bradley M. Spieler. "The Effect of Image Resolution on Deep Learning in Radiography". In: *Radiology: Artificial Intelligence* 2.1 (2020), e190015. DOI: 10.1148/ryai.2019190015. eprint: https://doi.org/10.1148/ryai.2019190015. URL: https://doi.org/10.1148/ryai.2019190015.

[84] Hojjat Salehinejad et al. "Generalization of Deep Neural Networks for Chest Pathology Classification in X-Rays Using Generative Adversarial Networks". In: *2018 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*. 2018, pp. 990–994. DOI: 10.1109/ICASSP.2018.8461430.

[85] Tim Salimans et al. "Improved Techniques for Training GANs". In: *Proceedings of the 30th International Conference on Neural Information Processing Systems*. NIPS'16. Barcelona, Spain: Curran Associates Inc., 2016, pp. 2234–2242. ISBN: 9781510838819.

[86] Hongming Shan et al. "3-D Convolutional Encoder-Decoder Network for Low-Dose CT via Transfer Learning From a 2-D Trained Network". In: *IEEE Transactions on Medical Imaging* 37.6 (June 2018), pp. 1522–1534. ISSN: 1558-254X. DOI: 10.1109/tmi.2018.2832217. URL: http://dx.doi.org/10.1109/TMI.2018.2832217.

[87] C. E. Shannon. "A mathematical theory of communication". In: *The Bell System Technical Journal* 27.3 (1948), pp. 379–423. DOI: 10.1002/j.1538-7305.1948.tb01338.x.

[88] Hoo-Chang Shin, Holger R. Roth, et al. *Deep Convolutional Neural Networks for Computer-Aided Detection: CNN Architectures, Dataset Characteristics and Transfer Learning*. 2016. arXiv: 1602.03409 [cs.CV].

[89] Hoo-Chang Shin, Neil A Tenenholtz, et al. *Medical Image Synthesis for Data Augmentation and Anonymization using Generative Adversarial Networks*. 2018. arXiv: 1807.10225 [cs.CV].

[90] Nripendra Kumar Singh and Khalid Raza. *Medical Image Generation using Generative Adversarial Networks*. 2020. arXiv: 2005.10687 [eess.IV].

[91] Matthew Stewart. *Comprehensive Introduction to Autoencoders*. Apr. 2019. URL: https://towardsdatascience.com/generating-images-with-autoencoders-77fd3a8dd368.

[92] Melissa Conrad Stöppler. *Medical Definition of Retinal fundus*. Mar. 2021. URL: https://www.medicinenet.com/retinal_fundus/definition.htm.

[93] Christian Szegedy et al. *Going Deeper with Convolutions*. 2014. arXiv: 1409.4842 [cs.CV].

[94] The pandas development team. *pandas-dev/pandas: Pandas*. Version latest. Feb. 2020. DOI: 10.5281/zenodo.3509134. URL: https://doi.org/10.5281/zenodo.3509134.

[95] Sergios Theodoridis. "Chapter 2 - Probability and Stochastic Processes". In: *Machine Learning (Second Edition)*. Ed. by Sergios Theodoridis. Second Edition. Academic Press, 2020, pp. 19–65. ISBN: 978-0-12-818803-3. DOI: https://doi.org/10.1016/B978-0-12-818803-3.00011-8. URL: https://www.sciencedirect.com/science/article/pii/B9780128188033000118.

[96] Daniel Vasata and Miroslav Cepek. *Neural networks*. https://courses.fit.cvut.cz/BIE-VZD/lectures/files/2020/11/BI-VZD-11-en-slides.pdf. Thákurova 9, 160 00 Prague 6: Department of Theoretical Computer Science , Faculty of Information Technology Czech Technical University in Prague, 2020.

[97] Linda Wang. *COVID-Net Open Source Initiative*. https://github.com/lindawangg/COVID-Net. 2020.

[98] Linda Wang, Zhong Qiu Lin, and Alexander Wong. "COVID-Net: a tailored deep convolutional neural network design for detection of COVID-19 cases from chest X-ray images". In: *Scientific Reports* 10.1 (Nov. 2020), p. 19549. ISSN: 2045-2322. DOI: 10.1038/s41598-020-76550-z. URL: https://doi.org/10.1038/s41598-020-76550-z.

[99] Eric W. Weisstein. *Convolution*. URL: https://mathworld.wolfram.com/Convolution.html (visited on 03/31/2021).

[100] Eric W. Weisstein. *Hyperbolic Functions*. URL: https://mathworld.wolfram.com/HyperbolicFunctions.html (visited on 04/21/2021).

[101] Eric W. Weisstein. *Metric*. URL: https://mathworld.wolfram.com/Metric.html (visited on 03/31/2021).

[102] Lilian Weng. *From GAN to WGAN*. 2019. arXiv: 1904.08994 [cs.LG].

[103] Jelmer M. Wolterink et al. "Deep MR to CT Synthesis Using Unpaired Data". In: *Simulation and Synthesis in Medical Imaging*. Ed. by Sotirios A. Tsaftaris et al. Cham: Springer International Publishing, 2017, pp. 14–23. ISBN: 978-3-319-68127-6.

[104] Michael Wooldridge and Nicholas R. Jennings. "Intelligent agents: theory and practice". In: *The Knowledge Engineering Review* 10.2 (1995), pp. 115–152. DOI: 10.1017/S0269888900008122.

[105] *X-Rays*. Apr. 2021. URL: https://medlineplus.gov/xrays.htm.

[106] Yuan Xue et al. "SegAN: Adversarial Network with Multi-scale L1 Loss for Medical Image Segmentation". In: *Neuroinformatics* 16.3-4 (May 2018), pp. 383–392. ISSN: 1559-0089. DOI: 10.1007/s12021-018-9377-x. URL: http://dx.doi.org/10.1007/s12021-018-9377-x.

[107] Dong Yang et al. *Automatic Liver Segmentation Using an Adversarial Image-to-Image Network*. 2017. arXiv: 1707.08037 [cs.CV].

[108] Xin Yi and Paul Babyn. "Sharpness-Aware Low-Dose CT Denoising Using Conditional Generative Adversarial Network". In: *Journal of Digital Imaging* 31.5 (Feb. 2018), pp. 655–669. ISSN: 1618-727X. DOI: 10.1007/s10278-018-0056-0. URL: http://dx.doi.org/10.1007/s10278-018-0056-0.

[109] Xin Yi, Ekta Walia, and Paul Babyn. "Generative adversarial network in medical imaging: A review". In: *Medical Image Analysis* 58 (Dec. 2019), p. 101552. ISSN: 1361-8415. DOI: 10.1016/j.media.2019.101552. URL: http://dx.doi.org/10.1016/j.media.2019.101552.

[110] Ha YongWook. *VAE Keras*. https://github.com/YongWookHa/VAE-Keras. 2018.

[111] *Your Rights Under HIPAA*. Nov. 2020. URL: https://www.hhs.gov/hipaa/for-individuals/guidance-materials-for-consumers/index.html.

[112] Simiao Yu et al. *Deep De-Aliasing for Fast Compressive Sensing MRI*. 2017. arXiv: 1705.07137 [cs.CV].

[113] Le Zhang, Ali Gooya, and Alejandro F. Frangi. "Semi-supervised Assessment of Incomplete LV Coverage in Cardiac MRI Using Generative Adversarial Nets". In: *Simulation and Synthesis in Medical Imaging*. Ed. by Sotirios A. Tsaftaris et al. Cham: Springer International Publishing, 2017, pp. 61–68. ISBN: 978-3-319-68127-6.

[114] He Zhao, Huiqi Li, and Li Cheng. *Synthesizing Filamentary Structured Images with GANs*. 2017. arXiv: 1706.02185 [cs.CV].

[115] Jun-Yan Zhu et al. *Unpaired Image-to-Image Translation using Cycle-Consistent Adversarial Networks*. 2020. arXiv: 1703.10593 [cs.CV].

Contents of an attached medium

- `readme.txt` ..... Brief overview of the medium
- `code`
  - `notebook` ..... Interactive notebooks
  - `src` ..... Source code of the implementation
  - `thesis` ..... Thesis' source code in LaTeX
- `text` ..... Thesis text
  - `thesis.pdf` ..... Thesis text in PDF format