Slovak University of Technology in Bratislava
Faculty of Informatics and Information Technologies

FIIT-5212-102868

Ladislav Bielko

Image Generating Encapsulated Environment

Bachelor's Thesis

Supervisor: Dr. Giang Nguyen Thu

May 2022

Slovak University of Technology in Bratislava
Faculty of Informatics and Information Technologies

FIIT-5212-102868

Ladislav Bielko

Image Generating Encapsulated Environment

Bachelor's Thesis

Study program: Informatics

Study field: B-INFO

Training workplace:
Institute of Informatics, Information Systems and Software Engineering

Supervisor: Dr. Giang Nguyen Thu

May 2022

Slovenská technická univerzita v Bratislave
Ústav informatiky, informačných systémov a softvérového inžinierstva

Fakulta informatiky a informačných technológií
Akademický rok: 2021/2022
Evidenčné číslo: FIIT-5212-102868

ZADANIE BAKALÁRSKEJ PRÁCE

Študent:	**Ladislav Bielko**
ID študenta:	102868
Študijný program:	informatika
Študijný odbor:	informatika
Vedúca práce:	Ing. Giang Nguyen Thu, PhD.
Vedúci pracoviska:	doc. Ing. Valentino Vranić, PhD.

Názov práce: **Zapuzdrené prostredie pre generovanie obrázkov (tvár alebo avatar)**

Jazyk, v ktorom sa práca vypracuje: slovenský jazyk

Špecifikácia zadania:

Zapuzdrené prostredie (angl. containerization) je dnes kritickým prvkom pre vývoj softvéru. Jeho použitie umožňuje vybudovať inteligentné aplikácie od prototypu k nasadeniu pohodlne a časovo efektívne. Momentálne, generative adversarial networks (GAN) sú relatívne novou témou v hlbokom učení, umožňujúcou generovanie obsahu s komplexnými črtami. Analyzujte súčasný stav kontajnerizácie softvéru a generovania údajov pomocou GAN. Trénujte inteligentný model na generovanie nových obrázkov (tvár alebo avatar) na základe verejne dostupných zdrojov. Ako praktickú ukážku vybudovania zapuzdreného prostredia, vytvorte Docker kontajner s aplikáciou pre dátovú vedu. Z vytrénovaného modelu realizujte flask aplikáciu na vizualizáciu. Vyhodnoťte výsledný softvérový produkt vrátane kvality vytvoreného modelu.

Rozsah práce: 40

Termín odovzdania bakalárskej práce:	16. 05. 2022
Dátum schválenia zadania bakalárskej práce:	23. 11. 2021
Zadanie bakalárskej práce schválil:	doc. Ing. Valentino Vranić, PhD. – garant študijného programu

I declare on my own that I have prepared this work independently, on the basis of consultations and using the mentioned literature.

In Bratislava, May 2022 Ladislav Bielko

Annotation

Slovak University of Technology in Bratislava
Faculty of Informatics and Information Technologies
Study programme: Informatics

Author: Ladislav Bielko
Diploma Thesis: Image Generating Encapsulated Environment
Supervisor: Ing. Giang Nguyen Thu, PhD.
May 2022

Generative adversarial networks - GANs - are a well-studied, novel type of deep learning model, primarily used for generating new content from existing samples. They consist of two separate neural networks, a generator and a discriminator.

The name of this type of deep learning model implies it incorporates an adversary - this adversary is one of the aforementioned neural networks, the discriminator. It is this neural network that is the key part of such a model and allows for incremental refinement of the generated results.

The two neural networks work as follows - the discriminator is a common deep convolutional neural network for classification. It recognizes whether an input sample - an image or other type of data on which we train the model - is real or fake, i.e., whether it comes from the original data set or has been created by the generator.

The generator itself is a kind of inverse deep convolutional neural network - it produces a new sample of content from a random numerical input, often using successive convolutions and up-scaling. In training, its goal is to fool the discriminator - to convince it that the sample it has generated is actually real.

By gradually training towards this goal, the generator's outputs gradually begin to take on the characteristics of real samples, eventually achieving outputs that are at the very least, at first glance, indistinguishable from real samples.

Anotácia

Slovenská technická univerzita v Bratislave
Fakulta informatiky a informačných technológií
Študijný program: Informatics

Autor: Ladislav Bielko
Diplomová práca: Zapúzdrené prostredie pre generovanie obrázkov
Vedúci diplomového projektu: Ing. Giang Nguyen Thu, PhD.
May 2022

Generative adversarial networks - GANs - sú dôkladne skúmaným, novým typom modelu hlbokého učenia, používaným najmä na generovanie nového obsahu z existujúcich vzoriek. Pozostávajú z dvoch samostatných neurónových sietí, generátora a diskriminátora.

Anglický názov tohto typu modelu hlbokého učenia - v preklade "generatívna sieť s protivníkom" naznačuje, že zahŕňa akéhosi protivníka - týmto protivníkom je jedna z uvedených neurónových sietí, diskriminátor. Práve táto neurónová sieť je kľúčovou súčasťou takéhoto modelu a umožňuje postupné vylepšovanie vygenerovaných výsledkov.

Generátor a diskriminátor fungujú nasledovne - diskriminátor je bežná hlboká konvolučná neurónová sieť na klasifikáciu. Rozpozná, či je vstupná vzorka - obrázok alebo iný typ údajov, na ktorých trénujeme model - skutočná alebo falošná, t. j. či pochádza z pôvodného súboru údajov, alebo bola vytvorená generátorom.

Samotný generátor je druh inverznej hlbokej konvolučnej neurónovej siete - vytvára novú vzorku obsahu z náhodného číselného vstupu, často pomocou postupných konvolúcií a zväčšovania. Pri tréningu je jeho cieľom oklamať diskriminátor - presvedčiť ho, že falošná vzorka, ktorú vytvoril, je skutočná.

Postupným trénovaním za týmto cieľom začnú výstupy generátora postupne nadobúdať vlastnosti skutočných vzoriek, až nakoniec dosiahne výstupy, ktoré sú prinajmenšom na prvý pohľad nerozoznateľné od skutočných vzoriek.

Acknowledgments

The acknowledgements and the people to thank go here, don't forget to include your project advisor…

Contents

Chapter 1

Introduction

Chapter 2

State of the art

The following is an analysis on the topic of machine learning, artificially generated data, deep learning models that generate that data, and tools for developing such models. The goal of this analysis is to gather information for creating our own deep learning model that generates artificial image data, an application that makes use of such, as well as a reliable means reproducing the training process for others.

2.1 Data

There are a plethora of things we experience in our everyday lives that hold value or information. Everything in our day-to-day life

2.1.1 Data from the real world

When seeking data, we usually look for information taken from the real world. We want real, authentic samples taken from the world by sensors, whether it be text, image, audio, or numerical data. We use this data to solve tasks like research,

2.1.2 Artificially generated data

Despite the enormous amounts of data found in the real world, we, however, may still want to produce artificial data. This may be the case for various reasons. We may not have enough real world data required for performing a specific task optimally. Artificially generated data is not always ideal for this, as we are still working within

the same scope of data - we are unlikely to find new information simply by generating new data similar to samples we already possess.

Another reason may be more personal. One may, as long as it falls under the terms of service of the platform, use an artificially generated image to represent themselves online. Like a mask, it serves the purpose of seeming believable, while protecting the user from their identity being revealed, where such an occurrence may suppose a risk.

2.2 Image processing with AI

The incentive to automate human-like content creation has led to various algorithms

However, the advantage of using deep learning for such tasks is that it brings creativity, arguably the most important part of creating human content, to the table. Although certainly not biologically accurate, neural networks implement the core concept of brain activity - neurons, minuscule computational units - interconnected and working together to create complex ideas. This concept has the power to, with enough training, create complex results out of a computationally simple task.

2.2.1 Deep learning

[1] [2]

2.2.1.1 Activation functions

2.2.1.2 Loss functions

2.2.2 Common types of deep learning models

2.2.2.1 Perceptrons

2.2.2.2 Recurrent neural networks

2.2.2.3 Convolutional neural networks

2.2.2.4 Long short-term memory networks

2.3 Content generation with deep learning

2.3.1 Transformers

Transformers are a type of deep learning model for processing and generating text or image data. They were proposed in a 2017 paper published by team consisting of researchers at Google[3]. Initially designed to handle natural language processing, transformers employ an attention-based system, rather than relying on convolutional or recurrent neural networks. Due to this, they are quite unlike other generative models.

2.3.1.1 GPT and its successors

[4] [5] [6]

2.3.1.2 DALL-E

[7]

2.3.2 Generative adversarial networks

Generative adversarial networks - GANs - are a well-studied, novel type of deep learning model, primarily used for generating new content from existing samples. They consist of two separate neural networks, a generator and a discriminator.

The name of this type of deep learning model implies it incorporates an adversary - this adversary is one of the aforementioned neural networks, the discriminator. It is this neural network that is the key part of such a model and allows for incremental refinement of the generated results.

The two neural networks work as follows - the discriminator is a common deep convolutional neural network for classification. It recognizes whether an input sample - an image or other type of data on which we train the model - is real or fake, i.e., whether it comes from the original data set or has been created by the generator.

The generator itself is a kind of inverse deep convolutional neural network - it produces a new sample of content from a random numerical input, often using successive convolutions and up-scaling. In training, its goal is to fool the discriminator - to convince it that the fake sample it has generated is actually real.

By gradually training towards this goal, the generator's outputs gradually begin to take on the characteristics of real samples, eventually achieving outputs that are at the very least, at first glance, indistinguishable from real samples.

The generator itself never sees the actual real samples from the dataset - it is only gradually nudged in a specific direction by its loss function (2.2.1.2), based on whether it failed to cause the discriminator to classify the generated sample as a real sample.

2.3.2.1 Use of GANs

Practical use of GAN models may not be immediately apparent, and these models may seem like no more than a pretty sight. However, there have been numerous exciting applications developed since their conception.

One use [8] [9]

[10]

2.3.2.2 Difficulties of training GAN models

2.3.3 Autoencoders

[11]

2.3.3.1 Advantages over GANs

2.3.3.2 A case for GANs

2.3.4 Latent space

With more complex models, however, such as our own model described in later chapters, the number of parameters defining a point in the latent space may be as high as 100 or more.

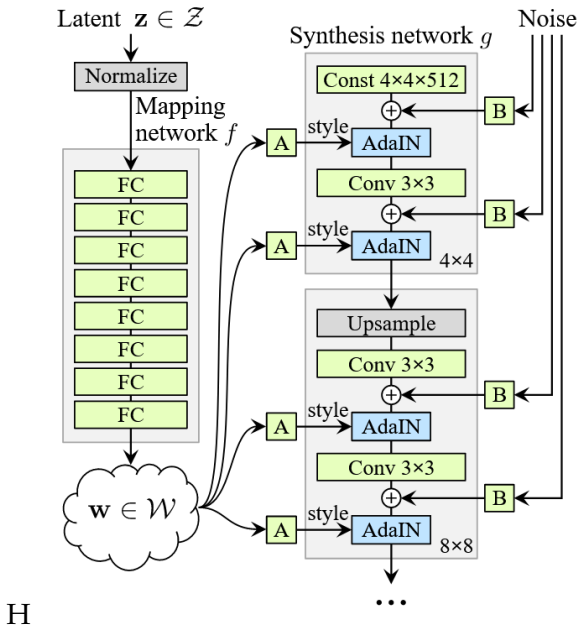

Figure 2.1: The structure of the StyleGAN generator model

2.4 State-of-the-art GAN models

Given that Generative adversarial networks are the focus of this analysis, we will go over state-of-the-art examples of these models in larger detail. Thus, these examples warrant a full section on their own.

2.4.1 BigGAN

[12]

2.4.2 StyleGAN and its successors

[13] [14] [15]

2.4.3 VQGAN + Clip

[16] [17]

2.5 Evaluation of generative models

When developing machine learning models, we want to see how well they perform, be it simply for statistics, or evaluating the results to see how the models could be improved. With simpler deep learning models, this is often an easy task - evaluating a ratio of how often the model made the correct prediction.

With image generating models, however, this task is much more complicated. Unlike simple deep learning models, such as basic dense layer or convolutional classifiers, this is not as simple as evaluating prediction accuracy, as GAN models consist of two separate neural networks.

2.5.1 The accuracy metric

If we do decide to evaluate a GAN model via prediction accuracy, we can only do so for the discriminator, meaning how well it can currently distinguish the real and fake generated samples. As for the generator, the task is a lot less straight forward, as we cannot get its accuracy directly.

2.5.2 Inception score

[18] [19] [20]

2.5.3 Fréchet Inception Distance

For this reason,

[21]

2.6 Development of data science systems

2.6.1 Containers

2.6.2 The Docker platform

2.6.3 Data science environments

2.7 Conclusion of analysis

Chapter 3

Objective

The objective of this thesis and the work within was to research current state-of-the-art technologies in image processing and generation, and to create our own image-generating deep learning model. We intended to focus the thesis on generative adversarial networks and aimed to broaden our knowledge in the field, while simultaneously also seeking knowledge for developing a higher quality model and meaningful experiments.

Another goal was to visualize these experiments in action, making use of a web application. Such visualisations would include traversal of the latent space in a manner that tries to explain the nature of the latent space of GANs. These visualisations were to be concise, comprehensive, yet interesting to those who may not have yet explored image generation and GANs to such depths as of yet.

Last but not least, we wanted a platform to design and develop the model in an effective and repreducable manner, by using Jupyter notebooks, Python documents that make works easily repreducable, due to their linear format. Furthermore, we aimed to make this Jupyter environment reliable and platform independent. This was to be achieved by encapsulating the environment in a container on the Docker platform.

Chapter 4

Methodology

For our analysis, and the thesis as a whole, we researched topics on deep learning, data processing, and generation thanks to books and articles availible on platforms such as ArXiv and Google Scholar.

For visualising the design of individual parts of our model, like the generator, discriminator, and their individual parts, as well as the training process, we used freely availible tools for diagram creation, found on the Diagrams.com platform.

As for our work itself, we based the initial design and development of our DCGAN model as described in the work [22]. Specifically, we started developing the model based on the grayscale MNIST generating model described in the book, working towards a functioning full color face-generating DCGAN model.

Chapter 5

Design

As part of this thesis, we designed and implemented our own GAN model[23] - a DCGAN, short for Deep convolutional generative adversarial network, a simple type of GAN that makes use of common convolutional neural network principles, while implementing multiple improvements, based on the original DCGAN paper[24].

We designed this model in Python 3, utilizing the Keras API, a wrapper for Tensorflow, designed specifically for creating deep learning models.[25]

5.1 The design of our DCGAN model

Our model consists of two convolutional neural networks - the generator and the discriminator. Both are created via the Keras *Sequential* class. These two models are then wrapped inside a main GAN model, itself being a *Sequential* class model.

The model is a GAN of variable length, meaning that both the generator and the discriminator have a variable number of blocks depending on the desired resolution of the images we want to generate.

5.1.1 Generator design

The generator network of our DCGAN model is a special type of convolutional neural network. Its structure is as follows:

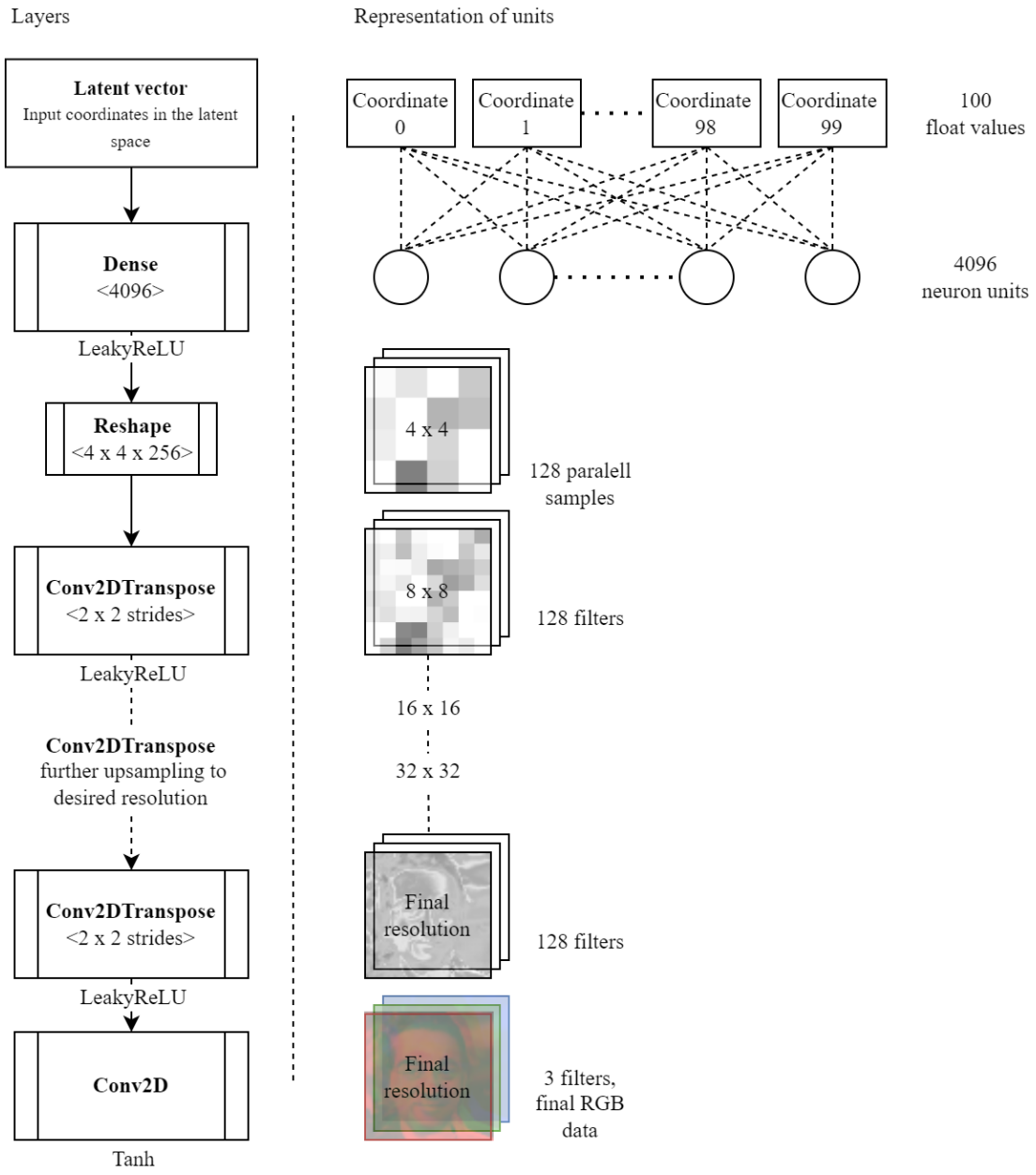

Figure 5.1: The structure of our DCGAN generator model

Figure 5.1 shows the structure of the layers present in the generator model, as well as visual representations of their outputs. We will go over the individual layers of the model in further detail.

The model takes a latent vector - a point in the latent space - as input. This point defines a unique face - or other subject the model may be trained on - in the latent space. For a large number of parameters that define each unique point, we have designed these vectors to be 100 in size. We examined the nature of latent spaces in further detail in section 2.3.4.

The rest of the following layers, aside from the final convolutional layer, are hidden layers. The *Reshape* layer takes the output of the previous layer - the output of each neuron in the *Dense* layer, and reshapes it into a 3D array of floating point values, thus creating a set of 256 two-dimensional arrays that are each 4 pixels wide in both dimensions. These serve as the initial stepping stone for the convolutional layers of the model.

The *Conv2DTranspose* layers are convolution layers with the ability to upsample. They each perform convolution filters on the previous layer. Combining upsampling and convolution, they create further details on each successive layer. The kernel size of these layers is 4x4, a multiple of the strides, as a 3x3 kernel causes checkerboard patterns on the output. This is different from normal convolution layers and is due to the padding during the upsampling.

The dense layer, as well as the individual upsample layers, are activated using the LeakyReLU activation function, allowing smaller negative values to pass through. The final convolution layer is activated using the hyperbolic tangent function - TanH.

As shown in the figure, the length of the model is variable depending on the desired final resolution, as each upsampling *Conv2DTranspose* layer doubles the resolution of the previous.

The base resolution we start with is always 4x4, as we found this to be an effective compromise of having good variation without needing further upsampling from 1x1 and 2x2. Smaller initial resolutions make the model more complex, lengthening the training process, as such is proportional to the number of neuron units in the model. Larger initial resolutions were found to not produce results that were as varied.

5.1.2 Discriminator design

The discriminator network of our DCGAN model is a convolutional neural network for image classification. Its structure is as follows:

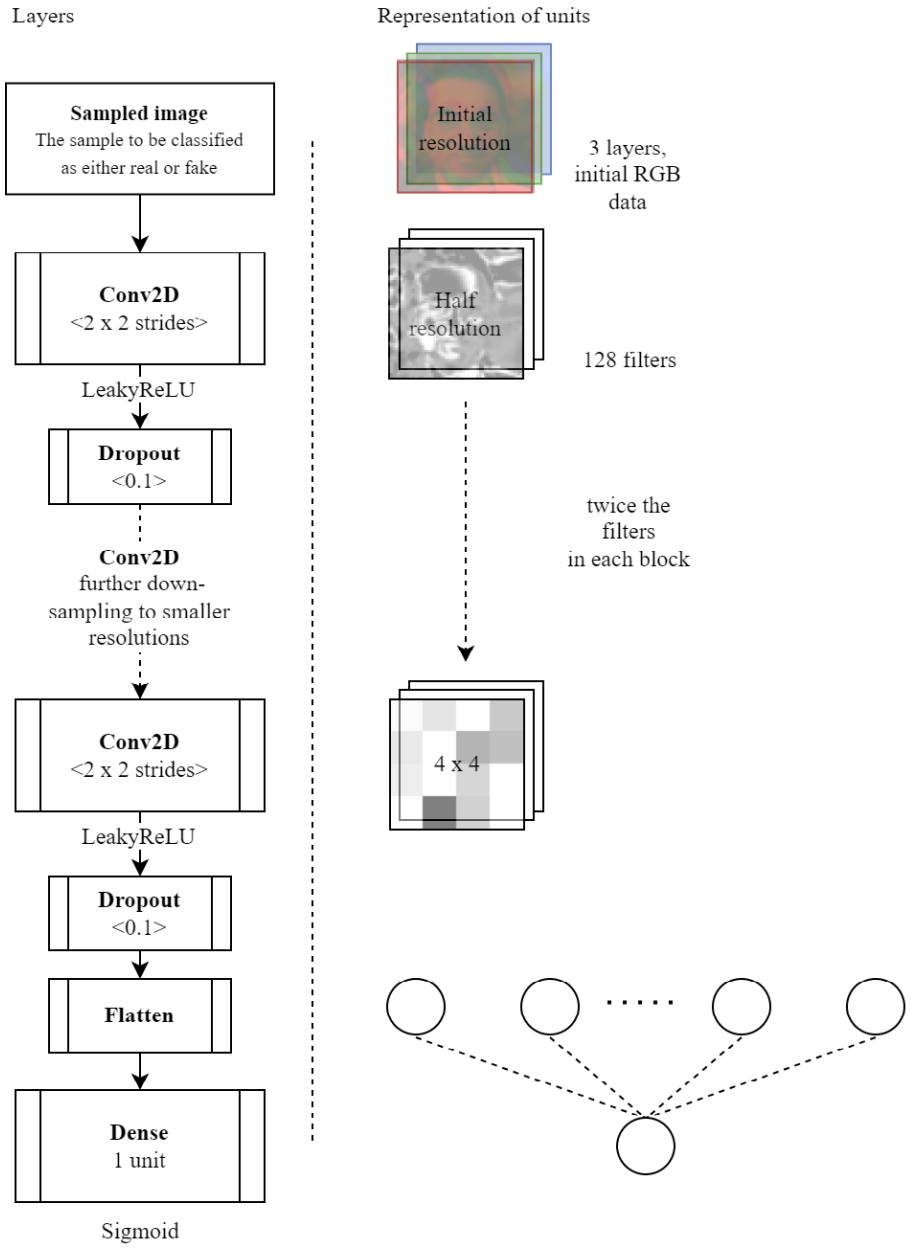

Figure 5.2: The structure of our DCGAN discriminator model

Figure 5.2 shows the layers present in the model, as well as visual representations of their outputs. We will go over the individual layers of the model in further detail.

The model takes a

Other convolutional neural network models employ pooling to downsample the resolution of the input. Our design instead does this downsampling directly in the convolution layers themselves, taking strides of two units in each dimension while applying the convolution filters. Each downsampled "pixel" still inherits the information of its surrounding pixels from the previous layer's output, similarly to max pooling, thanks to convolution filters with a kernel size of 3.

5.1.3 Training

The following is a description of the process we designed for training our generator and discriminator models.

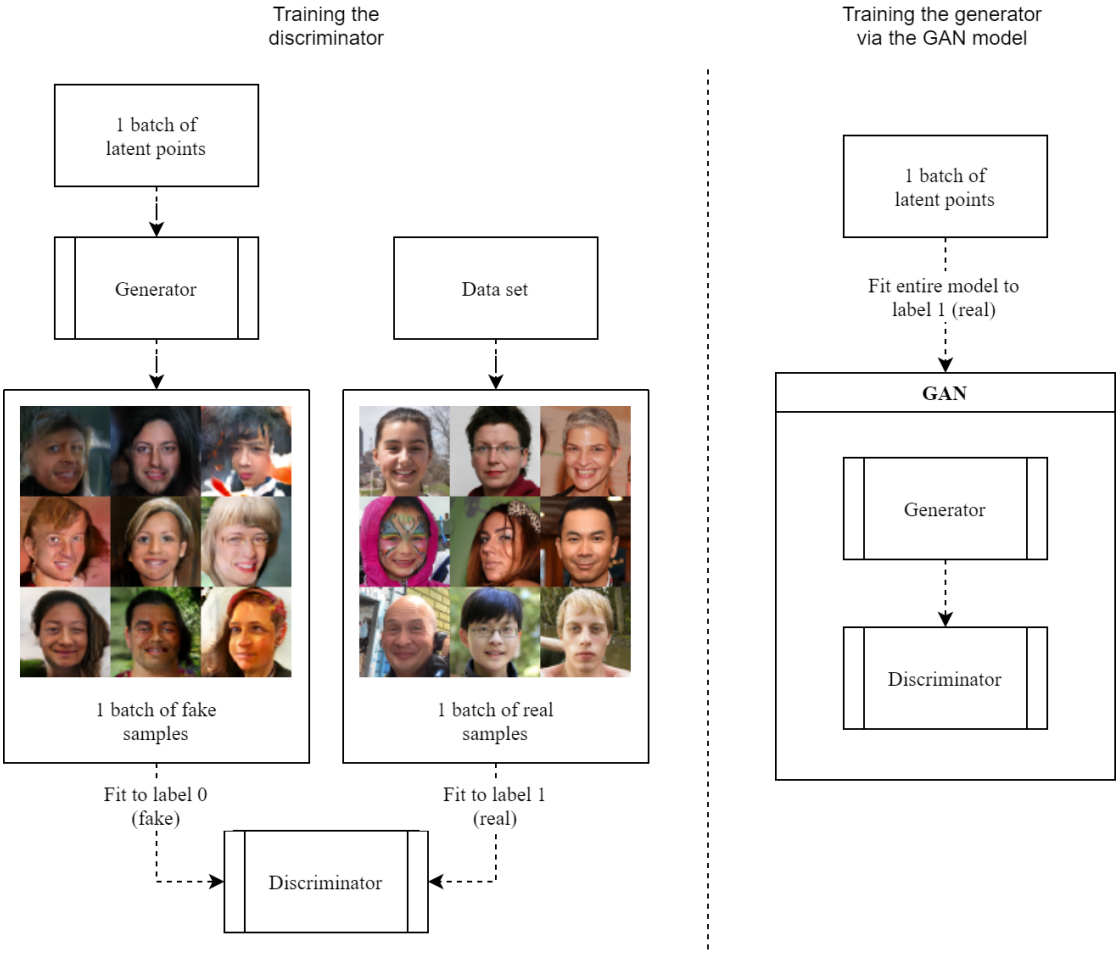

Figure 5.3: The process of training of our DCGAN model

Given that epochs are usually defined as the number of times a network was fitted to the entire dataset, we feed the dataset to the network sequentially, not by random chance, to ensure this property of each epoch. Indices of each image may be shuffled to avoid issues with this way of training, for example, with datasets that group similar images together.

5.2 Experiments and visualization

To test the usability of our model, as well as to showcase some interesting use cases for Generative Adversarial Networks (GAN)s, we designed multiple experiments described in detail below. The first two experiments are integral parts of our Flask application.

5.2.1 Embedding a real sample into the latent space

To showcase a real use for GANs, we designed a way of interpreting images that were not present in the training dataset into the latent space of our model. This allows, if the model is properly trained on a dataset of faces such as the Flickr-Faces-HQ (FFHQ) dataset, for a user to input their own face into the system, and get a latent space representation of it, with which they can perform various actions such as intuitively modifying some facial features and seeing the changes in real time, or interpolating their face with another, as described in the next experiment.

5.2.2 Latent space traversal

To visualize how the images generated by a GAN model are themselves not random, but rather depend on the latent vector sampled from the normal distribution, and that small changes to this vector result in similarly small changes in the output image, we wrote a Python function that takes the latent space representations of two different images, and creates a smooth transition between them.

This transition can be turned into an animated video by generating an image frame for each intermittent vector returned by the function, resulting in a unique morphing transformation between the two inputs. To achieve visual smoothness in this animated transition, we use the sine function for interpolating between each pair of values in the two input vectors, instead of a linear interpolation, the low point of the period being input A and the high point being input B.

As a side effect, this function may be used in a fun way to take the faces of two parents as input and return the latent vector situated in the very middle of the interpolation, resulting in an image of what their child may look like, as the facial features of the two parents are combined into one face.

5.2.3 Training our model on any Google search term

To showcase the generality of GANs, we designed an experiment with our DCGAN model that enables it to be trained on the type of image of our choice without the need of a premade dataset.

Optionally, if we do not want to train on faces, but rather on things like images of nature, the face cropping algorithm may be skipped, only cropping the entire image into a square aspect ratio so that the training images have a consistent shape and do not need to be padded.

5.2.4 Modifying our model for image impainting

Finally, we propose an image impainting experiment, utilizing our existing DCGAN model and modifying it in a way that creates a full image, impainting missing pixels from a patch in the input image. This may be achieved by modifying only the generator, having its input be an image with a patch of missing pixels, outputting a full image, and the discriminator remaining unchanged, differentiating between these filled-in images and existing ones from a given dataset.

To do this, the underlying architecture of the generator needs to be changed, incorporating elements of an autoencoder to encode the remaining pixels into a lower-dimensional representation, which will be easier to create upsampling convolutions from.

Chapter 6

Implementation

6.1 Designing and training the GAN model

6.1.1 Different iterations of the model

Throughout the development of our GAN model, we have gone through various iterations, starting from a small, 32-pixel GAN model, working up to a full 128-pixel DCGAN model.

6.2 The Flask web application

Chapter 7

Evaluation

7.1 Evaluation of the DCGAN model

7.1.1 Visual evaluation of outputs

In the 128 pixel version, the results at specific epochs were showing similar characteristics with the lower resolutions, albeit with added detail, but longer training times. Some samples, specifically those closer to the center of the normal distribution, were visually quite close to realistic, while those with more dispersed characteristics showed extensive artifacts and distortions. However, with extensive training over time, even those had improved, as can be seen in the figure below.

7.1.2 Evaluating the generator using FID

7.1.3 Impact on generator output after upgrading the DCGAN model

7.2 Experiment results

Chapter 8

Conclusion

Both the project and analysis parts of this thesis have been integral in getting to know the inner workings of deep learning and convolutional neural networks, as well as finding various incredible applications of GAN networks.

Due to previous experience with deep learning models and the scope of the project, the first submission of this thesis has been focused on designing the model itself and the experiments. While the literature, including that which the model was based on[22], has been read to a moderate degree, our findings about the state of the art in this field are still to be fully documented, as well as expanded upon.

Thus, for the final submission of this thesis, we want to focus on expanding the written analysis of the state-of-the-art techniques, as well as improving and adding onto the functionality of our model. We will continue to develop the remaining GAN experiments, the finalized docker container for training the model locally, as well as the flask application that will visualize the results in real time.

Resumé

10% in Slovak

List of Abbreviations

FFHQ	Flickr-Faces-HQ
GAN	Generative Adversarial Networks

List of Figures

List of Tables

Bibliography

[1] I. Goodfellow, Y. Bengio, and A. Courville. *Deep Learning*. Adaptive Computation and Machine Learning series. MIT Press, 2016. ISBN: 9780262035613. URL: https://books.google.se/books?id=Np9SDQAAQBAJ.

[2] Francois Chollet. *Deep learning with Python*. Simon and Schuster, 2021.

[3] Ashish Vaswani et al. "Attention Is All You Need". In: *CoRR* abs/1706.03762 (2017). arXiv: 1706.03762. URL: http://arxiv.org/abs/1706.03762.

[4] Alec Radford et al. "Improving language understanding by generative pre-training". In: (2018).

[5] Alec Radford et al. "Language models are unsupervised multitask learners". In: *OpenAI blog* 1.8 (2019), p. 9.

[6] Tom B. Brown et al. "Language Models are Few-Shot Learners". In: *CoRR* abs/2005.14165 (2020). arXiv: 2005.14165. URL: https://arxiv.org/abs/2005.14165.

[7] Aditya Ramesh et al. "Zero-shot text-to-image generation". In: *arXiv preprint arXiv:2102.12092* (2021).

[8] Raymond A. Yeh et al. *Semantic Image Inpainting with Deep Generative Models*. 2017. arXiv: 1607.07539 [cs.CV].

[9] Guilin Liu et al. "Image inpainting for irregular holes using partial convolutions". In: *Proceedings of the European Conference on Computer Vision (ECCV)*. 2018, pp. 85–100.

[10] Rameen Abdal, Yipeng Qin, and Peter Wonka. "Image2stylegan: How to embed images into the stylegan latent space?" In: *Proceedings of the IEEE/CVF International Conference on Computer Vision*. 2019, pp. 4432–4441.

[11] Diederik P Kingma and Max Welling. "Auto-encoding variational bayes". In: *arXiv preprint arXiv:1312.6114* (2013).

[12] Andrew Brock, Jeff Donahue, and Karen Simonyan. "Large Scale GAN Training for High Fidelity Natural Image Synthesis". In: *CoRR* abs/1809.11096 (2018). arXiv: 1809.11096. URL: http://arxiv.org/abs/1809.11096.

[13] Tero Karras, Samuli Laine, and Timo Aila. "A Style-Based Generator Architecture for Generative Adversarial Networks". In: *CoRR* abs/1812.04948 (2018). arXiv: 1812.04948. URL: http://arxiv.org/abs/1812.04948.

[14] Tero Karras et al. "Analyzing and Improving the Image Quality of StyleGAN". In: *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*. June 2020.

[15] Tero Karras et al. "Alias-free generative adversarial networks". In: *Advances in Neural Information Processing Systems* 34 (2021).

[16] Patrick Esser, Robin Rombach, and Bjorn Ommer. "Taming Transformers for High-Resolution Image Synthesis". In: *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*. June 2021, pp. 12873–12883.

[17] Alec Radford et al. "Learning transferable visual models from natural language supervision". In: *arXiv preprint arXiv:2103.00020* (2021).

[18] Christian Szegedy et al. "Going deeper with convolutions". In: *Proceedings of the IEEE conference on computer vision and pattern recognition*. 2015, pp. 1–9. DOI: 123456.

[19] Christian Szegedy et al. "Rethinking the inception architecture for computer vision". In: *Proceedings of the IEEE conference on computer vision and pattern recognition*. 2016, pp. 2818–2826.

[20] Tim Salimans et al. "Improved techniques for training gans". In: *Advances in neural information processing systems* 29 (2016), pp. 2234–2242. DOI: 1234567.

[21] Naresh Babu Bynagari. "GANs Trained by a Two Time-Scale Update Rule Converge to a Local Nash Equilibrium". In: *Asian Journal of Applied Science and Engineering* 8 (2019), pp. 25–34.

[22] J. Brownlee. *Generative Adversarial Networks with Python: Deep Learning Generative Models for Image Synthesis and Image Translation*. Machine Learning Mastery, 2019. URL: https://books.google.sk/books?id=YBimDwAAQBAJ.

[23] Ladislav Bielko. *DCGAN-Docker*. https://github.com/2021-FIIT-Bc-projects/DCGAN-Docker. 2021.

[24] Yang Yu et al. “Unsupervised representation learning with deep convolutional neural network for remote sensing images”. In: *International Conference on Image and Graphics*. Springer. 2017, pp. 97–108.

[25] François Chollet. *Keras*. `https://github.com/fchollet/keras`. Accessed 16 Dec 2021. 2015.

Appendix A

First Appendix

Appendix A

Work Schedule

A.1 Work Schedule for Bachelor's Thesis 1

1st-4th week	Consultations & finding related research
5th-6th week	Experimenting with the model
7th week	Working on the State of the art and Introduction chapters
8th-10th week	Further developing the model, Working on the Design, Implementation and Evaluation chapters
11th-12th week	Finalizing the currently prepared code and chapters, writing the Objective, Methodology and Conclusion chapters.

A.2 Work Schedule for Bachelor's Thesis 2

1st-2nd week	Consultations & designing API specification
3rd-6th week	Consultations & implementation of back-end
7th-9th week	Implementation of server
10th week	Consultations & implementation of front-end
11th-12th week	Finishing documentation & solution testing