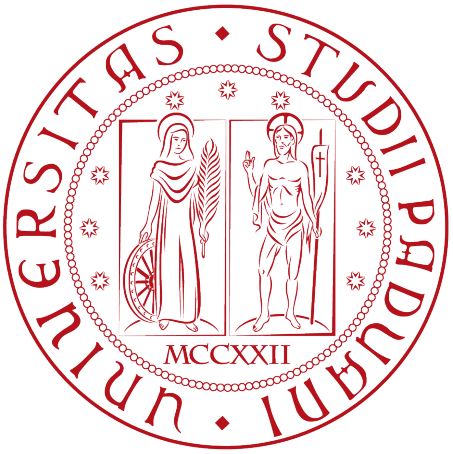
Università degli Studi di Padova

Department of Mathematics "Tullio Levi-Civita"

Degree Course in Computer Science



Generating Realistic Marble Textures using Generative Adversarial Networks

Thesis

Supervisor

Prof. Lamberto Ballan

Student

Marco Bernardi

Accademic Year 2022-2023

Summary

This document presents the outcomes of the internship conducted by Marco Bernardi at Breton S.p.A, spanning approximately three hundred hours. The internship encompassed various objectives, the first of which involved conducting a feasibility study on the utilization of Generative Adversarial Networks (GANs) for generating lifelike marble textures. Following the confirmation of feasibility, the subsequent goal was to develop a model capable of generating marble textures with a notable level of realism and diversity. Lastly, the final objective involved integrating the model into fundamental software designed for the creation of marble textures.

Contents

List of Figures

List of Tables

Chapter 1

Introduction

1.1 The company

Breton S.p.A is an Italian company specialized in the design, engineering and production of machinery and advanced systems for various industries. Established in 1963 by Marcello Toncelli, Breton gained international recognition as a leader in the production of cutting-edge industrial equipment. The company's headquarters are located in Castello di Godego, Italy, with numerous production facilities and branches strategically located around the world. Breton's experience spans multiple industries, including stone working, metal working, aerospace, automotive and more. Their extensive product portfolio includes a wide variety of machinery, catering for the needs of both small workshops and large industrial companies. With a strong focus on research and development, Breton has been instrumental in driving advances in automation and digitization within various industries. The company constantly invests in cutting-edge technologies such as the artificial intelligence, Internet of Things (IOT[g]) and machine learning to provide cutting-edge solutions to its customers. Also, Breton places a significant emphasis on sustainability and environmentally friendly practices. Through the development of energy efficient machinery, waste reduction initiatives and the promotion of sustainability production processes, Breton actively contributes to a greener and more sustainable environment future.

1.1.1 Products and Services

Products

Breton's product portfolio encompasses a wide range of machinery, including:

- **CNC machines:** Breton's CNC[g]machines are designed to provide high precision and accuracy in machining operations. The company offers a wide range of CNC machines, including vertical machining centers, horizontal machining centers, and 5-axis machining centers.
- **Cutting and shaping systems:** Breton's cutting and shaping systems are designed to provide high precision and accuracy in cutting and shaping operations. The company offers a wide range of cutting and shaping systems, including waterjet cutting systems, laser cutting systems, plasma cutting systems, and wire cutting systems.

- **Polishing equipment:** Breton's polishing equipment is designed to provide high precision and accuracy in polishing operations. The company offers a wide range of polishing equipment, including polishing machines, polishing robots, and polishing systems.
- **Robotic solutions:** Breton's robotic solutions are designed to provide high precision and accuracy in robotic operations. The company offers a wide range of robotic solutions, including robotic arms, robotic cells, and robotic systems.
- **Software:** Breton provides software solutions for the management of the productivity process, that can be fully integrated with the existing systems, for grant the best performance of the machinery and grant the control over the production plant.

Services

Breton offers a wide range of services to its customers, including:

- **Consulting:** Breton provides consulting services to its customers, helping them choose the right machinery for their needs. The company's experts analyze the customer's requirements and recommend the most suitable solutions.
- **Installation:** Breton's technicians install the machinery at the customer's site and ensure that it is functioning properly.
- **Training:** Breton offers training programs to its customers, teaching them how to operate the machinery and get the most out of it.
- **Maintenance:** Breton provides maintenance services to its customers, ensuring that the machinery is running smoothly and efficiently.

1.1.2 Certifications

Breton is continuously improving its products, services and workflows to meet the highest standards of quality, safety, and environmental protection.
The company is certified according to the following standards:

- **ISO 9001:** Breton is certified according to the ISO 9001 standard, which specifies requirements for a quality management system.
- **ISO 14001:** Breton is certified according to the ISO 14001 standard, which specifies requirements for an environmental management system.
- **UNI INAIL ed. 2001:** Breton is certified according to the UNI INAIL ed. 2001 standard, which specifies requirements for a health and safety management system.

1.2 The idea

The concept originated from a request made by the company aiming to extract key characteristics of a marble slab, including its veins, texture, and colors, in order to automatically generate a digital representation, or fingerprint, of the slab. This digital fingerprint can be effectively created using a conventional machine learning (ML[g])

model. However, accurately teaching the model to distinguish between veins and other features in marble slabs poses significant challenges, as the veins can vary and the slabs themselves are not always flawless.

To address this issue, a substantial number of images depicting the veins' paths are required to train the model effectively. However, obtaining such images is often problematic for various reasons:

- The company Breton primarily works with test slabs and thus lacks a large collection of images.
- Depending on customers for image contributions is not always feasible, as they may be unwilling to share their own images.

Manually extracting the veins' paths from images is an extremely time-consuming task that demands significant human resources. Consequently, the proposed solution involves creating a generative model capable of producing a substantial quantity of images that include the respective veins' paths. These generated images can then be utilized to train the machine learning model effectively.

1.2.1 Side Idea

Derived from the main concept, an ancillary idea emerged: the creation of a computer numerical control (CNC[g]) program based on an image file. This would enable the recreation of veins on engineered stone slabs using a CNC machine. However, implementing this idea presents considerable challenges, as converting an image file into a CNC program is a complex task. A CNC program consists of a sequence of instructions for the machine, while an image file is composed of a matrix of pixels.

Consequently, the ancillary idea was revised and transformed into the following: "Generating a marble slab image from a computer-aided design (CAD[g]) model". This approach proves more viable, as the CAD model can be readily converted into a CNC program, which in turn can be utilized to fabricate a marble slab.

To accomplish this, a generative adversarial network (GAN[g]) model is required. The GAN model must possess the ability to accurately reproduce colors, textures, and veins that adhere to the path defined by the CAD model. This approach offers increased feasibility, as the CAD model can be easily transformed into a CNC program, enabling the subsequent production of a marble slab.

1.3 Goals

The goals of the project can be categorized into three main categories denoted by the following letters:

- **M:** Mandatory goals, which represent the primary objectives of the project and are explicitly required by the commissioning company.
- **D:** Desirable goals, which serve as secondary objectives for the project and are not essential for the commissioning company but can add value to the overall outcome.
- **O:** Optional goals, which are additional objectives for the project that may be pursued based on feasibility and available resources.

Each goal is assigned a unique code consisting of the category letter followed by a number that identifies the specific objective. The goals are listed in the following table:

Table 1.1: Goals

Code	Description	Category
M1	Get an analytic and multidisciplinary thought, thanks to the decomposition of the problem in sub-problems	M
M2	Reach a level of autonomy in the management of the project, with synthesis and critical thinking of the problems	M
M3	Quality on the production of technological artifacts and on the documentation	M
D1	Development of a POC[g] for generating marble images with a GAN model or similar technology	D
D2	Development of a POC for augmenting resolution of an image	D
D3	Validation of the artifacts produced	D
O1	First product engineering approaches developed	O
O2	Product testing in a manufacturing production environment	O

1.4 Text structure

First chapter The first chapter introduce the company and its products and services. It also describes the idea of the project and the text structure.

Second chapter The second chapter describes the process and the methodology used for the development of the project.

Third chapter Third chapter describes the internship experience, and how the project was planned and developed.

Fourth chapter The fourth chapter describes how the project technologies were implemented for reach the objectives.

Fifth chapter The fifth chapter describes how the project was tested and validated.

Sixth chapter The sixth chapter describes the conclusions of the project.

During the writing of this document, the following conventions were adopted:

- acronyms, abbreviations and ambiguous or uncommon terms mentioned are defined in the glossary, located at the end of this document;
- for the first occurrence of the terms defined in the glossary, the following nomenclature is used: *word*[g];
- the terms that require further explanation like technical words are marked with a subscript *italic*;

Chapter 2

Process and methodologies

In this chapter, we delve into the comprehensive exploration of the technologies employed during the internship and their broader application.

2.1 GANs

The concept of Generative Adversarial Networks (GAN) was first introduced by Ian Goodfellow and his colleagues in 2014 in their paper titled "Generative Adversarial Networks"[1] published at the Neural Information Processing Systems (NIPS) conference. Goodfellow, along with his co-authors, proposed a novel framework that revolutionized the field of generative modeling. GANs are a groundbreaking approach to generative modeling that combines elements of both supervised and unsupervised learning. They consist of two interconnected neural networks: a generator and a discriminator. The generator network learns to generate new data samples, such as images or text, by mapping random input vectors to output samples that resemble the training data. The discriminator network, on the other hand, aims to distinguish between real data samples from the training set and those generated by the generator. These two networks engage in a competitive process, where the generator strives to produce samples that the discriminator cannot differentiate from real data, while the discriminator aims to correctly classify the samples. Through iterative training, GANs are able to improve the quality of the generated samples, leading to increasingly realistic and high-fidelity outputs. GANs have since become a cornerstone in the field of generative modeling and have found applications in various domains, including image synthesis, text generation, and video generation.

2.2 Train process

The training process of a GAN model involves a unique adversarial framework that iteratively improves the generator and discriminator networks. Initially, both networks are randomly initialized. During training, the generator takes random input vectors and generates synthetic data samples. Simultaneously, the discriminator receives both real data samples from the training set and generated samples from the generator. The

[1] I. Goodfellow et al. "Generative Adversarial Networks". In: *Advances in Neural Information Processing Systems 27* (), pp. 2672–2680. URL: http://papers.nips.cc/paper/5423-generative-adversarial-nets.pdf.

discriminator's objective is to accurately distinguish between real and fake samples, while the generator aims to produce samples that can fool the discriminator into classifying them as real. The training process occurs in alternating steps. In each step, the discriminator is trained by optimizing its parameters to minimize the classification error, correctly identifying real and generated samples. Conversely, the generator is trained by adjusting its parameters to maximize the error rate of the discriminator, essentially trying to generate samples that are indistinguishable from real data. This adversarial game continues for multiple iterations, with the generator and discriminator networks continuously updating their weights to improve their performance. Through this iterative process, the generator learns to produce increasingly realistic samples, while the discriminator becomes more adept at distinguishing between real and generated data. The training process of a GAN is complex and requires careful balancing. If the generator becomes too powerful, it may produce samples that closely resemble the training data but lack diversity. On the other hand, if the discriminator becomes too strong, it can easily detect generated samples, resulting in poor-quality outputs. Achieving a delicate equilibrium between the two networks is essential for training a successful GAN model.

2.3 Evaluation process

2.3.1 The problem of evaluating GANs

In contrast to conventional deep learning models that are trained with a loss function until convergence, Generative Adversarial Networks (GANs) operate within a zero-sum game framework involving two interconnected networks: the generator and the discriminator. The generator aims to deceive the discriminator by generating realistic samples, while the discriminator endeavors to accurately differentiate between genuine and fake samples. The training process concludes when the discriminator becomes incapable of distinguishing between real and synthetic samples, signifying that the generator has successfully captured the underlying distribution of the training data.

This unique training approach of GANs poses a significant challenge when it comes to objective evaluation and assessment. Unlike traditional models that have objective functions to minimize or maximize, GANs lack a definitive metric for gauging training progress and determining the absolute or relative performance of a GAN model solely based on loss. This presents several complexities in various scenarios, such as selecting a final GAN model during training, showcasing the capabilities of a GAN through generated samples, comparing different GAN models, or comparing different hyperparameters for the same GAN model.

To date, the most prevalent approach for evaluating GANs involves a combination of qualitative and quantitative metrics that center around the quality and diversity of the generated samples.

2.3.2 Manual evaluation

Manual evaluation often serves as a means of assessing GAN models through the visual inspection of generated samples. This evaluation method relies on human judgment to gauge the quality of a batch of generated samples, rendering it subjective in nature. Although manual inspection is a straightforward approach to model evaluation, it entails several limitations. Firstly, subjectivity is introduced due to the evaluator's biases towards the model and the data. Secondly, expertise in the specific domain of the

data is required to effectively evaluate the samples. Furthermore, manual evaluation is time-consuming, imposing constraints on the number of images that can be thoroughly assessed.

. . . evaluating the quality of generated images with human vision is expensive and cumbersome, biased [. . .] difficult to reproduce, and does not fully reflect the capacity of models[2].

Consequently, while manual evaluation provides an initial impression of a model's performance, it should not be solely relied upon for the final selection of a model. Fortunately, alternative and more objective evaluation methods have been proposed and embraced within the field.

2.3.3 Qualitative evaluation

Qualitative evaluation plays a crucial role in assessing the visual quality and performance of GAN models. While subjective in nature, it provides valuable insights into various aspects of the generated samples, including their fidelity, coherence, diversity, and novelty. Several commonly used metrics and evaluation techniques have been developed to facilitate qualitative assessment.

- **Nearest neighbors**: Nearest neighbors evaluation involves comparing the generated samples with real samples from the training dataset. By computing the similarity between the generated samples and their nearest neighbors in the real data space, evaluators can assess how well the GAN model captures the underlying distribution of the training data;
- **Rapid Scene Categorization**[3]: aims to evaluate the ability of GAN-generated samples to be quickly recognized and categorized by human observers. Evaluators assess how well the generated samples align with the expected scene categories and their visual characteristics. This evaluation metric provides insights into the semantic coherence and overall discriminability of the generated samples;
- **Rating and Preference judgement**[4][5][6][7]: involve human evaluators rating or ranking the quality of generated samples based on predefined criteria. This qualitative evaluation method provides subjective assessments of visual quality, realism, and aesthetic appeal. By gathering ratings or preference judgments from multiple evaluators, a collective assessment of the generated samples can be obtained;
- **Mode Drop and Collapse**[8][9]: Mode drop and collapse refer to situations where

[2] Ali Borji. "Pros and Cons of GAN Evaluation Measures". In: *arXiv preprint arXiv:1802.03446* abs/1802.03446 (). URL: https://arxiv.org/abs/1802.03446.

[3] Ibid.

[4] Xun Huang et al. "Stacked Generative Adversarial Networks". In: *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)* (). URL: https://arxiv.org/abs/1612.04357.

[5] Han Zhang et al. "StackGAN: Text to Photo-realistic Image Synthesis with Stacked Generative Adversarial Networks". In: *IEEE International Conference on Computer Vision (ICCV)* (). URL: https://arxiv.org/abs/1612.03242.

[6] C. Xiao et al. "Generating adversarial examples with adversarial networks". In: *arXiv preprint arXiv:1801.02610* (). URL: https://arxiv.org/abs/1801.02610.

[7] Z. Yi et al. "DualGAN: Unsupervised Dual Learning for Image-to-Image Translation". In: *arXiv preprint arXiv:1704.02510* (). URL: https://arxiv.org/abs/1704.02510.

[8] A. Srivastava et al. "Veegan: Reducing Mode Collapse in GANs using Implicit Variational Learning". In: *Advances in Neural Information Processing Systems* (). URL: https://arxiv.org/abs/1705.07761.

[9] Z. Lin et al. "PacGAN: The power of two samples in generative adversarial networks". In: *arXiv*

the GAN model fails to generate samples representing all the diverse modes or aspects of the training data distribution. Evaluators visually inspect the generated samples to identify any mode drop, where certain modes or patterns are missing, or mode collapse, where the generated samples lack diversity and exhibit repetitive patterns. Assessing mode drop and collapse is crucial for evaluating the ability of GAN models to capture the full range of variations in the training data;

- **Network Internals**[10][11][12][13][14][15]: Analyzing the internal representations and activations of the GAN model's neural network can provide insights into the learning process and the generated samples' quality. Evaluators examine network internals, such as feature maps and intermediate layers, to gain a better understanding of how the GAN model generates and captures visual patterns;

The most used qualitative measure is a sort of manual inspection of images, called *Rating and Preference judgement.*

. . . These types of experiments ask subjects to rate models in terms of the fidelity of their generated images. . . [16].

Usually images are shown in pairs (one real and one fake), and the subjects are asked to choose the best image. A score or rating is then assigned to the model based on the number of times it is chosen as the best image. For lowering the variance of the results, the images are shown to multiple human judges and the results are averaged. This process is labor intensive, but with the help of crowd-sourcing platforms like Amazon Mechanical Turk, it can be done at scale (Reducing the cost also).
Another downside of this method is that the human judges performance can improve with experience, especially if they are given feedback on their performance.

. . . By learning from such feedback, annotators are better able to point out the flaws in generated images, giving a more pessimistic quality assessment. . . [17].

---

preprint arXiv:1712.04086 (). URL: https://arxiv.org/abs/1712.04086.

[10] A. Radford, L. Metz, and S. Chintala. "Unsupervised Representation Learning with Deep Convolutional Generative Adversarial Networks". In: *arXiv preprint arXiv:1511.06434* (). URL: https://arxiv.org/abs/1511.06434.

[11] X. Chen et al. "InfoGAN: Interpretable Representation Learning by Information Maximizing Generative Adversarial Nets". In: *Advances in Neural Information Processing Systems* (). URL: https://arxiv.org/abs/1606.03657.

[12] I. Higgins et al. "beta-VAE: Learning Basic Visual Concepts with a Constrained Variational Framework". In: (). URL: https://arxiv.org/abs/1711.00464.

[13] M. F. Mathieu et al. "Disentangling Factors of Variation in Deep Representation using Adversarial Training". In: *Advances in Neural Information Processing Systems* (). URL: https://arxiv.org/abs/1611.03383.

[14] M. D. Zeiler and R. Fergus. "Visualizing and Understanding Convolutional Networks". In: *European Conference on Computer Vision* (). URL: https://arxiv.org/abs/1311.2901.

[15] D. Bau et al. "Network Dissection: Quantifying Interpretability of Deep Visual Representations". In: *Computer Vision and Pattern Recognition* (). URL: https://arxiv.org/abs/1704.05796.

[16] Borji, "Pros and Cons of GAN Evaluation Measures".

[17] Ibid.

2.3.4 Quantitative evaluation

In addition to qualitative evaluation, quantitative assessment provides a systematic and objective analysis of GAN models' performance. These evaluation metrics aim to measure various aspects of the generated samples, including their diversity, fidelity, and similarity to the real data distribution. By quantitatively evaluating GAN models, researchers can compare different models, assess the impact of hyperparameters, and track progress during training. Some of the most common quantitative measures are: Average log-likelihood, Coverage Metric, Inception Score, Modified Inception Score, Mode Score, AM Score, Fréchet Inception Distance, Maximum Mean Discrepancy, The Wasserstein Distance, Birthday Paradox Test, Classifier Two-Sample Tests, Classification Performance, Boundary Distortion, Number of Statistically-Different Bins, Image Retrieval Performance, Generative Adversarial Metric, Tournament Win Rate, Normalized Relative Discriminative Score, Adversarial Accuracy and Adversarial Divergence, Geometric Score, Reconstruction Score, Image Quality measures, Low-level Image Statistics, Precision, Recall and F1 Score.

The most used quantitative measures are ***Inception Score*** and ***Fréchet Inception Distance***.

Inception Score

The Inception Score (IS[g]) is a widely used quantitative metric for evaluating the quality and diversity of generated samples in GAN models. It was proposed in 2016 by Tim Salimans et al. in *"Improved Techniques for Training GANs"*[18], provides a measure of both sample quality and class diversity. The IS is computed by first obtaining the predicted class probabilities for each generated sample using an Inception-v3 pre-trained classifier. Then, the average of these probabilities is calculated to assess the quality of the generated samples. Additionally, the entropy of the predicted class probabilities is computed to measure the diversity of the samples. The formula for calculating the Inception Score is as follows:

$$\mathrm{IS} = \exp\left(\mathbb{E}_{\mathbf{x}\sim p_{\mathrm{data}}}\left[\mathrm{KL}\left(p(y|\mathbf{x})||p(y)\right)\right]\right) \tag{2.1}$$

The KL divergence is a measure of how one probability distribution is different from a second, reference probability distribution. The KL divergence is defined as:

$$\mathrm{KL}(P||Q) = \sum_i P(i)\log\left(\frac{P(i)}{Q(i)}\right) \tag{2.2}$$

The Inception Score is a good measure of the quality of generated images, but it has some limitations. It is sensitive to dataset and classifier choice, lacks consideration of spatial coherence, may not detect mode collapse effectively, offers limited interpretability, and emphasizes high-quality samples. Complementary metrics and qualitative assessments are essential for a comprehensive evaluation of GAN models.

Frechet Inception Distance

The Fréchet Inception Distance (FID[g]) score is a widely used metric for evaluating the quality of generated images in the field of generative adversarial networks (GANs)

[18]Tim Salimans et al. "Improved Techniques for Training GANs". In: *arXiv preprint arXiv:1606.03498* abs/1606.03498 (). URL: http://arxiv.org/abs/1606.03498.

introduced by Martin Heusel et al.[19].

"FID performs well in terms of discriminability, robustness and computational efficiency [...] It has been shown that FID is consistent with human judgments and is more robust to noise than IS".[20]

It measures the similarity between the distribution of real images and the distribution of generated images by comparing their feature representations extracted from a pre-trained Inception-v3 network[21]. A lower FID score indicates better similarity between the two distributions, suggesting higher-quality generated images that resemble the real data more closely. The FID score takes into account both the quality and diversity of generated images, making it a valuable metric for assessing the performance of GAN models. It provides a quantitative measure that complements visual inspection and subjective evaluation, enabling researchers to objectively compare and analyze different GAN architectures and training strategies. The FID score is defined as:

$$\text{FID}(p, q) = \|\mu_p - \mu_q\|^2 + \text{Tr}(\Sigma_p + \Sigma_q - 2(\Sigma_p\Sigma_q)^{1/2}) \tag{2.3}$$

Where:

- μ_p and μ_q are the mean vectors of the real and generated images respectively.
- Σ_p and Σ_q are the covariance matrices of the real and generated images respectively.
- Tr is the trace operator.
- $\|\cdot\|$ is the Euclidean norm.

As the IS score, the FID score has some limitations based on the use of the Inception-v3 network.

Suggested GAN evaluation procedure

The evaluation of Generative Adversarial Networks (GANs) presents challenges, necessitating a comprehensive approach that combines qualitative and quantitative measures. Initially, a manual inspection of the generated images is recommended to assess the quality of the generator model. Subsequently, quantitative measures such as the Inception Score and the Frechet Inception Distance can be employed to evaluate the quality and diversity of the generated images. It is important to note that there is no universally superior measure for GAN evaluation, as the selection of evaluation measures depends on the specific task and dataset at hand.

As of yet, there is no consensus regarding the best score. Different scores assess various aspects of the image generation process, and it is unlikely that a single score can cover all aspects. Nevertheless, some measures seem more plausible than others (e.g. FID score)[22]

[19] Martin Heusel et al. "GANs Trained by a Two Time-Scale Update Rule Converge to a Local Nash Equilibrium". In: *arXiv preprint arXiv:1706.08500* abs/1706.08500 (). URL: http://arxiv.org/abs/1706.08500.

[20] Borji, "Pros and Cons of GAN Evaluation Measures".

[21] C. Szegedy et al. "Rethinking the Inception Architecture for Computer Vision". In: *Computer Vision and Pattern Recognition* (). URL: https://arxiv.org/abs/1512.00567.

[22] Borji, "Pros and Cons of GAN Evaluation Measures".

Chapter 3

Internship description

This chapter provides an overview of the internship project, including its requirements, goals, and the planning undertaken for the internship period.

3.1 Initial analysis

The initial phase of the project involved an analysis of the requirements for developing a functional GAN model. This process entailed posing pertinent questions to determine the desired outcomes and the appropriate approach. These inquiries encompassed the nature of the generated images, the input image requirements, the network architecture, the extracted image features, and the dataset selection. By addressing these questions, the project's requirements, goals, and a comprehensive roadmap were established, laying the foundation for subsequent project execution.

3.2 Requirements & Goals

3.2.1 Requirements

- **Requirement 1**: The dataset used for training the GAN model must consist of high-quality images of marble slabs to ensure accurate representation;
- **Requirement 2**: The model should possess the capability to generate realistic marble slab images based on given inputs, such as sketches or other forms of guidance;
- **Requirement 3**: The generated images produced by the model should exhibit a high level of quality, comparable to real images of marble slabs, in terms of visual fidelity, texture, and details;
- **Requirement 4**: The generated images should be presented in real time, allowing for immediate visual feedback and preview during the image generation process.

3.2.2 Goals

- **Goal 1**: Acquire a diverse and high-quality dataset of marble slab images that encompasses various patterns, colors, and textures to train the GAN model effectively;

- **Goal 2**: Develop a GAN model that demonstrates the ability to generate accurate and visually appealing marble slab images from provided inputs, such as sketches or other relevant data;
- **Goal 3**: Create a desktop application that employs the trained GAN model to generate marble slab images in real time, enabling users to preview the generated images promptly during the image generation process. This application should provide an intuitive user interface for seamless interaction.

3.3 Planning

Initially, the project was planned using a story map (See fig. 3.1) to define the main features of the project and the main steps to achieve them.

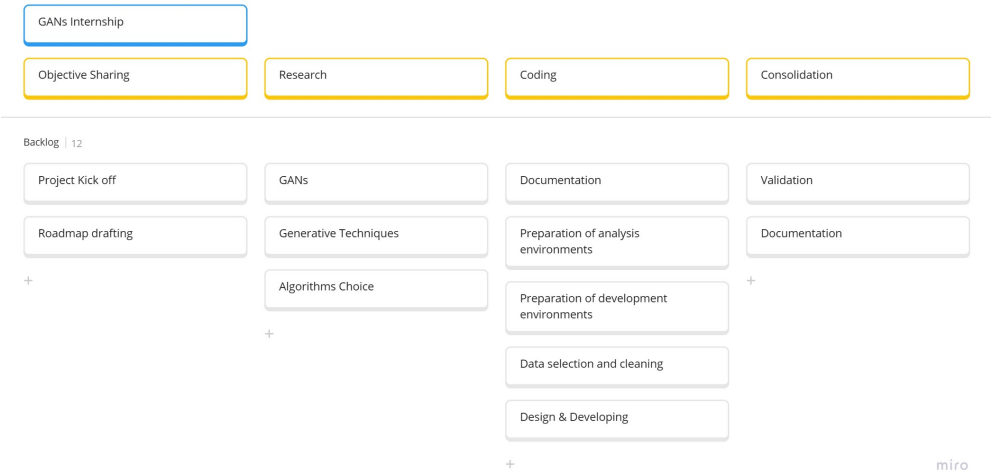


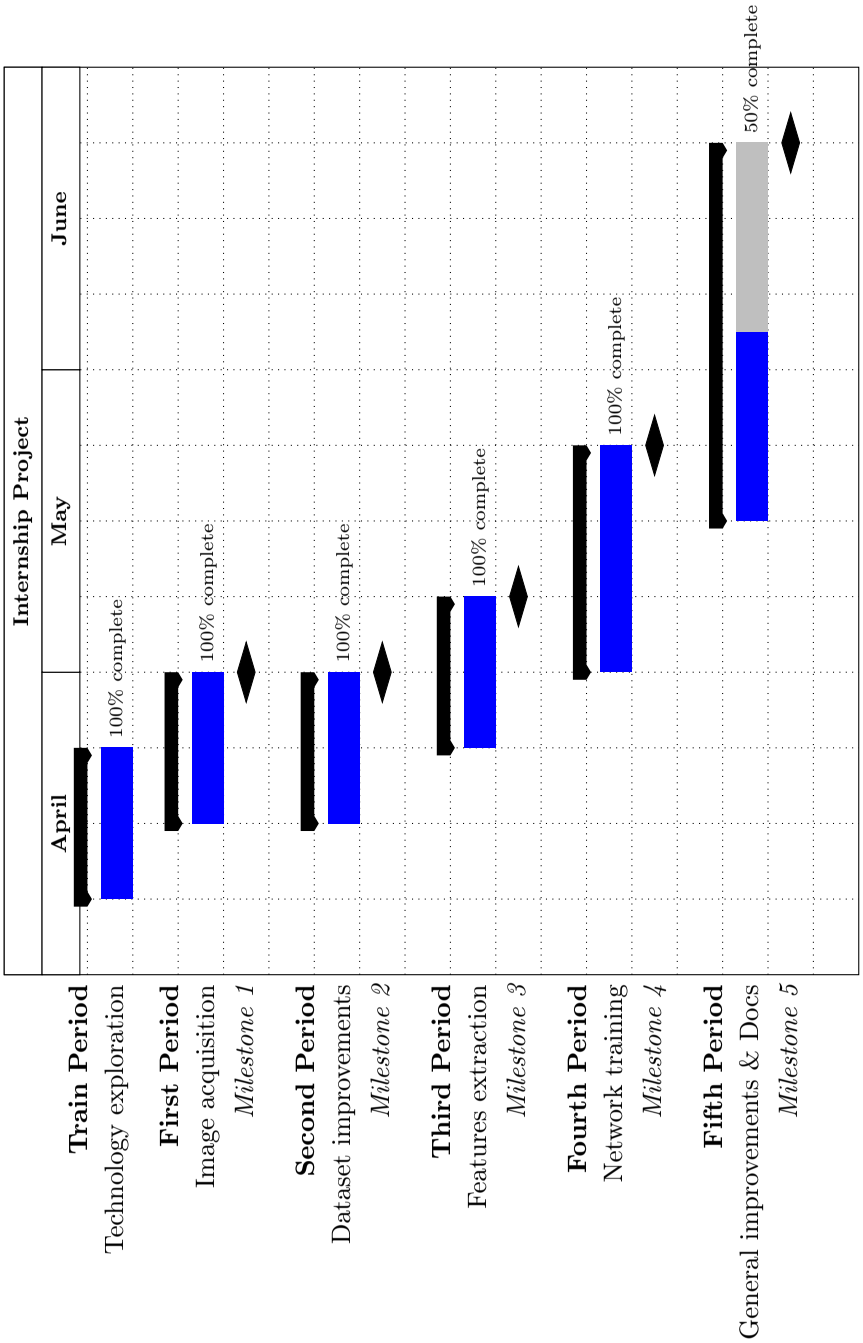
Figure 3.1: Story map

3.3.1 Road-map

Following an initial analysis of the story map, it was feasible to establish a project roadmap encompassing the requirements and objectives to be accomplished during the internship period. The entire undertaking has been meticulously planned, employing a distinct temporal division across various phases. These phases are outlined below:

- **Train period**: This phase entails an in-depth study of the current state of the art and the technologies that will be employed;
- **First period**: During this phase, the focus will be on image acquisition;
- **Second period**: The primary objective of this period is to augment the dataset by increasing the number of images;
- **Third period**: In this phase, the key task is to extract the main features;
- **Fourth period**: Network training constitutes the central objective of this Internship;

- **Fifth period**: The final phase involves the analysis of results and the implementation of improvements;


Internship Project
April
May
June
Train Period
Technology exploration
100% complete
First Period
Image acquisition
100% complete
Milestone 1
Second Period
Dataset improvements
100% complete
Milestone 2
Third Period
Features extraction
100% complete
Milestone 3
Fourth Period
Network training
100% complete
Milestone 4
Fifth Period
General improvements & Docs
50% complete
Milestone 5

3.3.2 Study Period:

This designated period was allocated for an in-depth examination of the current state of the art and the technologies that would be employed throughout the course of the internship. The primary focus areas encompassed the following key topics:

- **Machine Learning**: A comprehensive study of the fundamental concepts and principles underpinning machine learning and deep learning;
- **GAN**: An extensive exploration of the core concepts and functioning principles of GANs;
- **GAN applications**: An examination of the principal applications of GANs and their practical utilization;
- **GAN architectures**: A detailed analysis of the primary GAN architectures and their respective applications;
- **GAN training**: A thorough investigation into the predominant techniques employed for training GANs;
- **GAN evaluation**: An in-depth exploration of the leading methodologies utilized for evaluating the performance of GANs;
- **GAN improvements**: A comprehensive study of the key techniques employed to enhance and refine GAN models;
- **GAN applications**: An exploration of the primary domains where GANs find application and their practical implementation;

3.3.3 First Period:

During this designated period, the focus was on acquiring images. Each Breton machine was already equipped with a camera and a computer featuring specialized software capable of capturing images from the camera. The camera captured photographs of each worked slab, which were then transmitted to the database. Subsequently, the images were retrieved from the database and stored in a designated folder.

Challenges:

The primary challenge encountered during this phase pertained to the quality of the acquired images. The machine's image acquisition functionality was not originally intended for use in a machine learning project, resulting in suboptimal image quality. The images displayed variations in size, inconsistent backgrounds, and occasionally contained unwanted elements such as light reflections or machine shadows.

Solutions:

TTo address these challenges, the following measures were implemented:

- **Background Removal**: To eliminate extraneous elements from the images, a pre-trained machine learning model developed by Breton was employed. This model, in conjunction with the OpenCV library, effectively identified and removed non-slab components from the images, focusing solely on the slabs themselves.

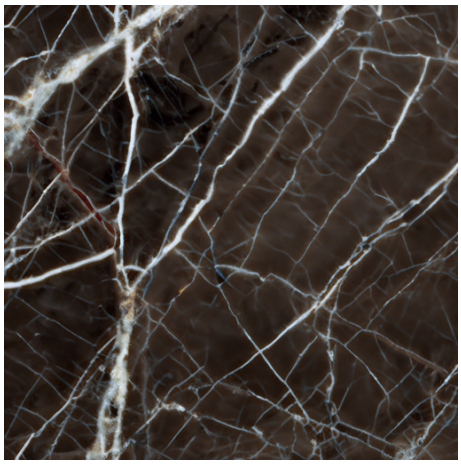


Figure 3.2: Example of a slab

- **Image Defects**: Manual intervention was employed to remove images with light reflections and shadows from the dataset. This task was feasible due to the manageable quantity of such images.
- **Images Resizing**: The images were resized to a standardized dimension while preserving the original aspect ratio.

Milestone:

At the conclusion of this period, the dataset consisted of 500 images of slabs (See fig. 3.2), featuring diverse colors and textures. The dataset was subsequently divided into two separate parts: one for training purposes and the other for validation, with a split ratio of 70% for training and 30% for validation, respectively.

3.3.4 Second Period:

During this designated period, the dataset was expanded, and the existing images underwent augmentation techniques.

Image Augmentation:

Image augmentation is a technique employed to increase the number of images in a dataset by applying various transformations to the original images. This technique proves valuable when the dataset size is insufficient to train a network effectively. To expand the dataset, each original image was split into smaller images of dimensions 256×256 pixels (See fig. 3.3). After that the following augmentation techniques were applied to the images:

- **Resize**: Each image was resized to a larger dimension, increasing from 256×256 pixels to 286×286 pixels;
- **Random Crop**: A random cropping operation was performed on each image, resulting in a standardized size of 256×256 pixels;
- **Random Flip**: Each image underwent a random horizontal flipping operation;

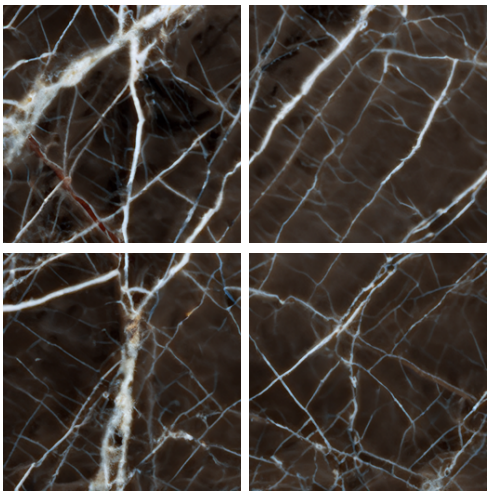


Figure 3.3: Cropped image

Milestone:

Upon the conclusion of this period, the dataset comprised 7000 images of slabs, exhibiting diverse colors and textures.

3.3.5 Third Period:

This designated period was dedicated to identifying the optimal method for extracting features from the images, such as veins, textures, and colors. The extraction of features from the images is a critical aspect of the project, as the quality of these features directly impacts the quality of the network. Thus, it was imperative to ensure the accuracy of the mask containing the extracted features.

Vein Extraction:

Various Python-implemented methods were tested for vein extraction. The tested methods included:

- **Thresholding**[1]: This method utilizes the Threshold algorithm, which applies a threshold to the image, retaining only the pixels with values higher than the threshold. To account for varying light conditions across images, the threshold was calculated using the Otsu's method.
- **Canny Edge Detection**[2]: This method employs the Canny algorithm, which is an edge detection algorithm utilizing a multi-stage approach to identify edges in images. The algorithm comprises five steps: noise reduction through Gaussian filtering, gradient calculation, non-maximum suppression, double thresholding, and edge tracking by hysteresis. Figure 3.4 illustrates the outcome of Canny Edge Detection

[1] *OpenCV: Thresholding Operations*. URL: https://docs.opencv.org/4.x/d7/d4d/tutorial_py_thresholding.html.

[2] *OpenCV: Canny Edge Detection*. URL: https://docs.opencv.org/3.4/da/d22/tutorial_py_canny.html.

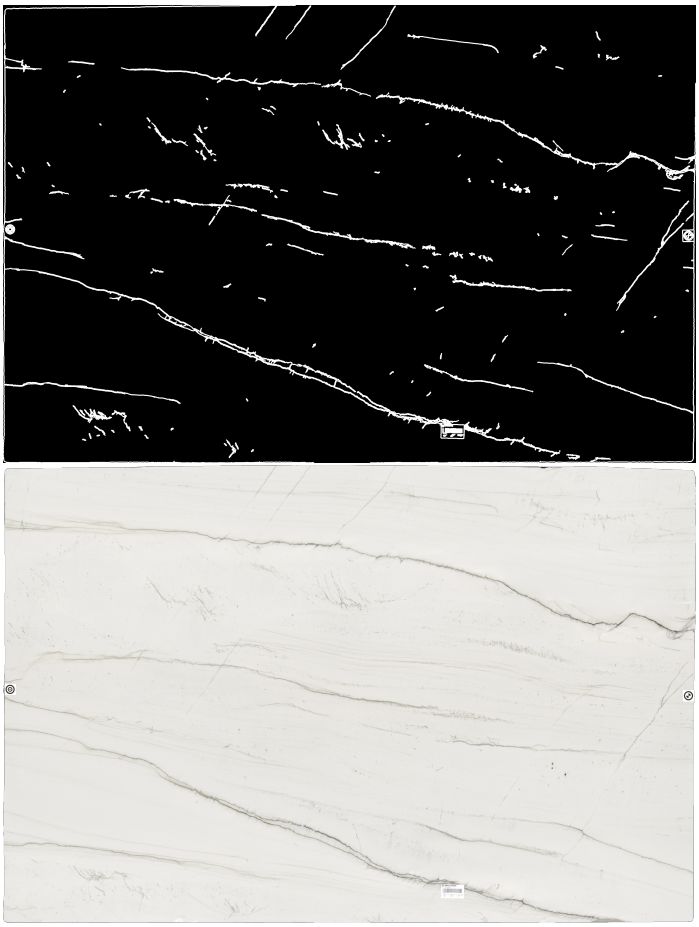


Figure 3.4: Canny Edge Detection

- **Meijering & Contrast filter**[3]: The Meijering filter is based on the Hessian matrix, which calculates the second-order partial derivatives of a function. Eigenvalues and eigenvectors obtained from the Hessian matrix enable the identification of line-like structures in the image. The contrast filter is employed to enhance image contrast. Figure 3.5 presents the result of the Meijering & Contrast filter.

[3] *Meijering Filter*. URL: https://scikit-image.org/docs/stable/api/skimage.filters.html#rbd62388c4e81-1.

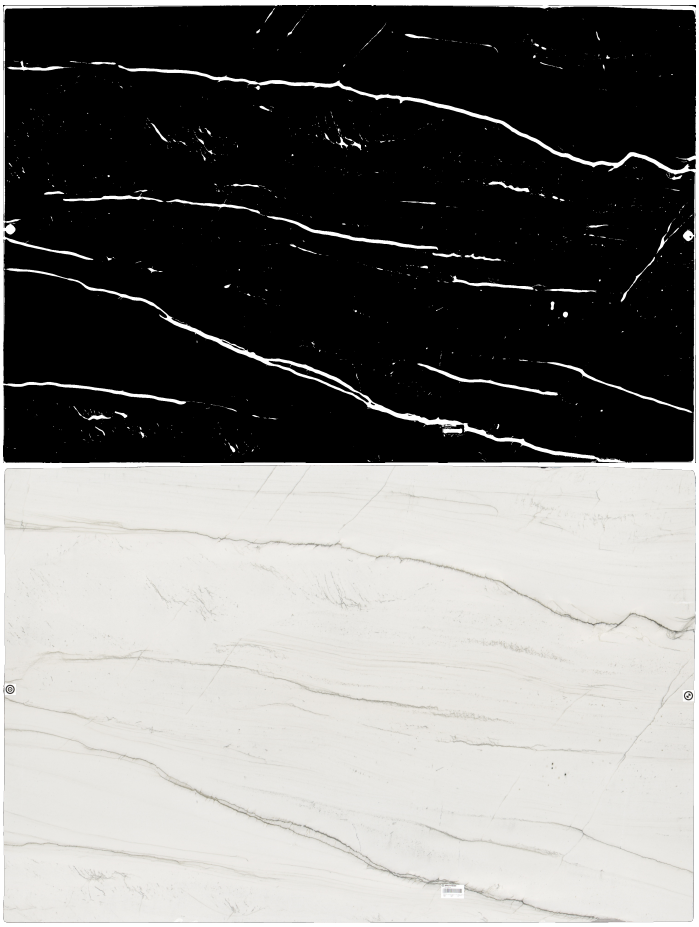


Figure 3.5: Meijering & Contrast filter

- **HED (Holistically-Nested Edge Detection)**[4]: This method is based on the HED[g]algorithm, which employs a deep neural network for edge detection in images. The algorithm involves three steps: utilizing a pre-trained network to extract features from the image, applying a multi-scale algorithm to extract edges from the features, and linking the edges using the Canny algorithm. Figure 3.6 showcases the result of the HED algorithm.

[4]S. Xie and Z. Tu. "Holistically-Nested Edge Detection". In: *International Conference on Computer Vision* (). URL: https://arxiv.org/abs/1504.06375.

Figure 3.6: HED

Milestone:

At the conclusion of this period, the results indicated that the Meijering & Contrast filter and the Canny Edge Detection methods were the most effective. The HED method was discarded due to its sluggish performance and subpar results, often introducing noise in the form of random lines into the images. The two preferred methods successfully extracted the vein structures from the images, considering that an unsupervised approach was employed.

Improvements:

One potential avenue for improvement involves the utilization of a supervised method for vein extraction. This method would offer greater accuracy compared to unsupervised methods, as it would be trained specifically for vein extraction. However, implementing a supervised approach necessitates significant time and a substantial number of hand-labeled images to train the model effectively.

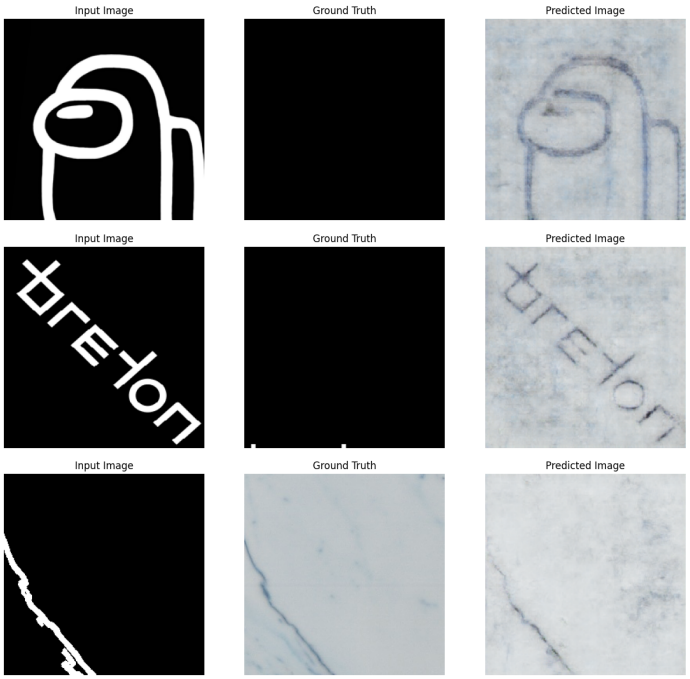


Figure 3.7: Generated images

3.3.6 Fourth Period:

During this designated period, the focus was on training the network. The network employed for this purpose was a pix2pix network, which utilizes a Conditional Generative Adversarial Network (CGAN) to generate images. The training process involved utilizing the dataset created in the preceding periods and the hardware resources provided by the company (See 4.1.8)

Various configurations of the network were tested during this period, involving the fine-tuning of hyperparameters to identify the optimal setup. The specific configuration details of the network will be elaborated upon in Chapter 4.

Milestone:

Upon the conclusion of this period, the network demonstrated its capability to generate images of slabs exhibiting a diverse range of colors and textures (See 3.7).

Chapter 4

Design and coding

This chapter covers the project's design and coding, including the technologies and tools used, the software life cycle, and the coding phase.

4.1 Technology and tools

In the subsequent sections, we will elucidate the technologies and tools employed within the project.

4.1.1 Python

Python is a high-level programming language renowned for its simplicity, readability, and versatility. It offers a clean and concise syntax, making it easy to understand and write code. Python's extensive standard library and thriving community contribute to its vast ecosystem of third-party libraries and frameworks, empowering developers to accomplish a wide range of tasks efficiently. From web development and scientific computing to data analysis and machine learning, Python excels in various domains. Its object-oriented nature, dynamic typing, and automatic memory management contribute to its flexibility and ease of use. Furthermore, Python's cross-platform compatibility enables code portability across different operating systems. With its consistent updates and improvements, Python continues to evolve, ensuring its relevance in the ever-changing landscape of software development.

4.1.2 CUDA

CUDA[g](Compute Unified Device Architecture) is a parallel computing platform and API model developed by NVIDIA. It allows developers to harness the computational power of NVIDIA GPUs[g]for general-purpose computing. By utilizing CUDA, developers can offload computationally intensive tasks to the GPU, resulting in significant performance improvements. CUDA provides a wide range of libraries and tools for efficient GPU programming, making it a standard for GPU computing in various industries and research fields. With ongoing advancements, CUDA continues to empower developers to leverage GPU parallelism for faster and more efficient computations.

4.1.3 CuDNN

CuDNN[g](CUDA Deep Neural Network) is a GPU-accelerated library developed by NVIDIA for deep learning tasks. It provides optimized implementations of key neural network operations, such as convolutions and recurrent operations. By leveraging CuDNN, developers can accelerate deep learning workflows, reducing training times and improving model performance. It seamlessly integrates with popular deep learning frameworks and supports mixed-precision training. CuDNN plays a crucial role in advancing deep learning research and applications by harnessing the power of GPU acceleration.

4.1.4 ML libraries

TensorFlow

TensorFlow is an open-source machine learning framework developed by Google. It provides a robust platform for building and deploying machine learning models. With its computational graph abstraction, TensorFlow enables efficient execution of complex mathematical computations. It offers a wide range of APIs and tools for different tasks, including model development, training, and deployment. TensorFlow supports distributed computing, allowing for parallel processing across multiple devices. It integrates with various frameworks and libraries, making it versatile and interoperable. TensorFlow is highly regarded for its scalability, flexibility, and extensive feature set, making it a popular choice among machine learning practitioners.

PyTorch

PyTorch is an open source machine learning library based on the Torch library, used for applications such as computer vision and natural language processing, primarily developed by Facebook's AI Research lab (FAIR). It is free and open source software released under the Modified BSD license. PyTorch serves as the foundation for several deep learning software applications, including Tesla Autopilot, Uber's Pyro, Hugging Face's Transformers, PyTorch Lightning, and Catalyst. It offers two prominent high-level capabilities: tensor computing, akin to NumPy, with enhanced acceleration through GPU utilization, and deep neural networks constructed on a tape-based automatic differentiation system. These features enable efficient computation and facilitate the development of sophisticated machine learning models.

Keras

Keras is a sophisticated, Python-based high-level neural networks API that can be seamlessly integrated with popular deep learning framework such as TensorFlow. It is specifically designed to facilitate rapid experimentation, allowing researchers and developers to quickly transition from conceptualizing ideas to obtaining meaningful results. The framework's core principle is to minimize the time lag between ideation and outcome, thereby facilitating efficient and effective research endeavors. Prominent scientific organizations worldwide, including CERN, NASA, NIH, and others, rely on Keras for their research and applications.

OpenCV

OpenCV, short for Open Source Computer Vision, is a popular and widely-used open-source computer vision library. It was initially developed by Intel in 1999 and later supported by Willow Garage and Itseez (now merged with Intel) before becoming a community-driven project. OpenCV provides a vast collection of computer vision algorithms and tools, making it a go-to solution for developers and researchers working on various vision-related tasks.

The primary goal of OpenCV is to provide a comprehensive and efficient infrastructure for computer vision applications. It supports a wide range of programming languages, including C++, Python, Java, and MATLAB, making it accessible to developers from diverse backgrounds. OpenCV offers a rich set of functions for image and video processing, feature detection and extraction, object recognition, camera calibration, and more.

One of the key strengths of OpenCV is its ability to leverage hardware acceleration, such as utilizing multicore CPUs and GPUs, to enhance performance. This makes it suitable for real-time and resource-intensive applications, such as robotics, augmented reality, surveillance systems, and autonomous vehicles.

The library encompasses a wide range of functionalities, including face detection and recognition, object identification, human action classification in videos, camera movement tracking, object motion tracking, 3D model extraction, generation of 3D point clouds from stereo cameras, image stitching for creating high-resolution panoramic images, similarity search in image databases, red-eye removal from flash photography, eye movement tracking, scene recognition, and marker establishment for augmented reality overlays, among others.

4.1.5 GANs Models

Pix2Pix

Pix2Pix[1] is a popular deep learning model used for image-to-image translation tasks. It is based on a conditional generative adversarial network (GAN) architecture, which consists of a generator network and a discriminator network. The Pix2Pix model aims to learn a mapping between an input image and an output image, where the output image is a transformed version of the input image according to a specific target domain.

The model was trained and evaluated on a large dataset of paired images from the *Berkeley Segmentation Dataset and Benchmark* and demonstrates a capability to generate plausible synthetic images for a variety of image-to-image translation tasks, such as converting daylight images to night.

Pix2Pix has been successfully applied to various image translation tasks, such as converting grayscale images to color, generating realistic street scenes from semantic labels, transforming sketches into photorealistic images, and more.

StyleGAN

StyleGAN is a highly versatile Generative Adversarial Network (GAN) model primarily developed for the purpose of general image generation. Extensive training and evaluation of the model were conducted using a substantial dataset comprised of unpaired images sourced from the Flickr-Faces-HQ dataset. Through this rigorous training

[1] Phillip Isola et al. "Image-to-Image Translation with Conditional Adversarial Networks". In: *arXiv preprint arXiv:1611.07004* abs/1611.07004 (). URL: http://arxiv.org/abs/1611.07004.

process, StyleGAN exhibits the ability to generate synthetic images that possess a remarkable level of plausibility across a wide range of image generation tasks. Notably, the model excels in the generation of realistic human faces, showcasing its proficiency in capturing the intricate details and characteristics associated with facial features.

4.1.6 Tools

Visual Studio Code

Visual Studio Code is a free source-code editor made by Microsoft for Windows, Linux and macOS. Features include support for debugging, syntax highlighting, intelligent code completion, snippets, code refactoring, and embedded Git.

Git

Git is a distributed version-control system for tracking changes in source code during software development. It is designed for coordinating work among programmers, but it can be used to track changes in any set of files. Its goals include speed, data integrity, and support for distributed, non-linear workflows.

GitHub

GitHub is a global company that provides hosting for software development version control using Git. It is a subsidiary of Microsoft, which acquired the company in 2018 for $7.5 billion. It offers all of the distributed version control and source code management (SCM) functionality of Git as well as adding its own features. It provides access control and several collaboration features such as bug tracking, feature requests, task management, and wikis for every project.

GIMP

GIMP is a free and open-source raster graphics editor used for image retouching and editing, free-form drawing, converting between different image formats, and more specialized tasks.

4.1.7 Adobe Photoshop

Adobe Photoshop, a widely recognized raster graphics editor, has been developed and published by Adobe Inc. for the Windows and macOS platforms. Since its inception in 1988 by Thomas and John Knoll, Photoshop has established itself as the industry standard for raster graphics editing and digital art creation. In a recent update, Adobe introduced a notable feature called "Generative Fill". This feature incorporates a Generative AI model, enabling users to generate new content within an image, thereby expanding the creative possibilities and augmenting the editing capabilities of the software.

Webex

Webex is a video conferencing software developed by Cisco Systems. It is a cloud-based software that provides video conferencing, online meetings, screen-sharing, and webinars. It has a free version that allows up to 100 participants, with a 50-minute time

restriction. The paid version starts at $13.50/month and allows up to 200 participants and unlimited meeting time.

Outlook

Outlook is a personal information manager software system from Microsoft, available as a part of the Microsoft Office suite. Primarily an email application, it also includes a calendar, task manager, contact manager, note taking, journal, and web browsing.

4.1.8 Hardware

The company provided all the necessary hardware for the development of this project. The hardware configuration utilized during the project is outlined as follows:

- **DELL Precision 7670**
  - **CPU**: Intel Core i7-12850HX
  - **GPU**: NVIDIA Quadro RTX A2000 8GB GDDR6
  - **RAM**: 32GB DDR4
  - **Storage**: 512GB NVMe SSD
  - **OS**: Windows 10 Pro 64-bit
- **DELL Precision 7520**
  - **CPU**: Intel Core i7-6820HQ
  - **GPU**: NVIDIA Quadro M2200 4GB GDDR5
  - **RAM**: 16GB DDR4
  - **Storage**: 512GB NVMe SSD
  - **OS**: Windows 10 Pro 64-bit

4.2 Final implementation

4.2.1 Network Type

In the ultimate implementation of the project, the selected network type was the Pix2Pix model.

Pix2Pix in detail

Pix2Pix is a comprehensive Generative Adversarial Network (CGAN) model specifically designed for the purpose of image-to-image translation. The Pix2Pix model comprises two distinct models, which are constructed as follows:

U-Net Generator

The generator follow the U-Net architecture, which is a convolutional neural network that consists of an encoder (down-sampler) and a decoder (up-sampler). The encoder downsamples the input image and extracts the features, while the decoder upsamples the image and produces the segmentation map. The skip connections between the

encoder and decoder are added to prevent the loss of low-level features during the upsampling process.

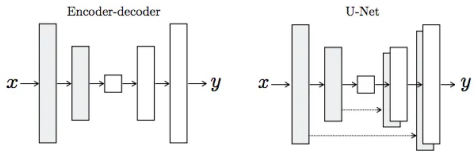


Figure 4.1: U-Net architecture

TensorFlow implementation

The TensorFlow implementation of the U-Net generator is composed by the following layers (see figure 4.2):

- **Encoder**: 8 downsampling layers, each downsampling layer is composed by a convolutional layer, a batch normalization layer, and a Leaky ReLU activation layer.
- **Decoder**: 8 upsampling layers, each upsampling layer is composed by a transposed convolutional layer, a batch normalization layer, a dropout layer(applied to the first 3 layers), and a ReLU activation layer.
- **Skip connections**: between the encoder and decoder, there are skip connections, each skip connection.

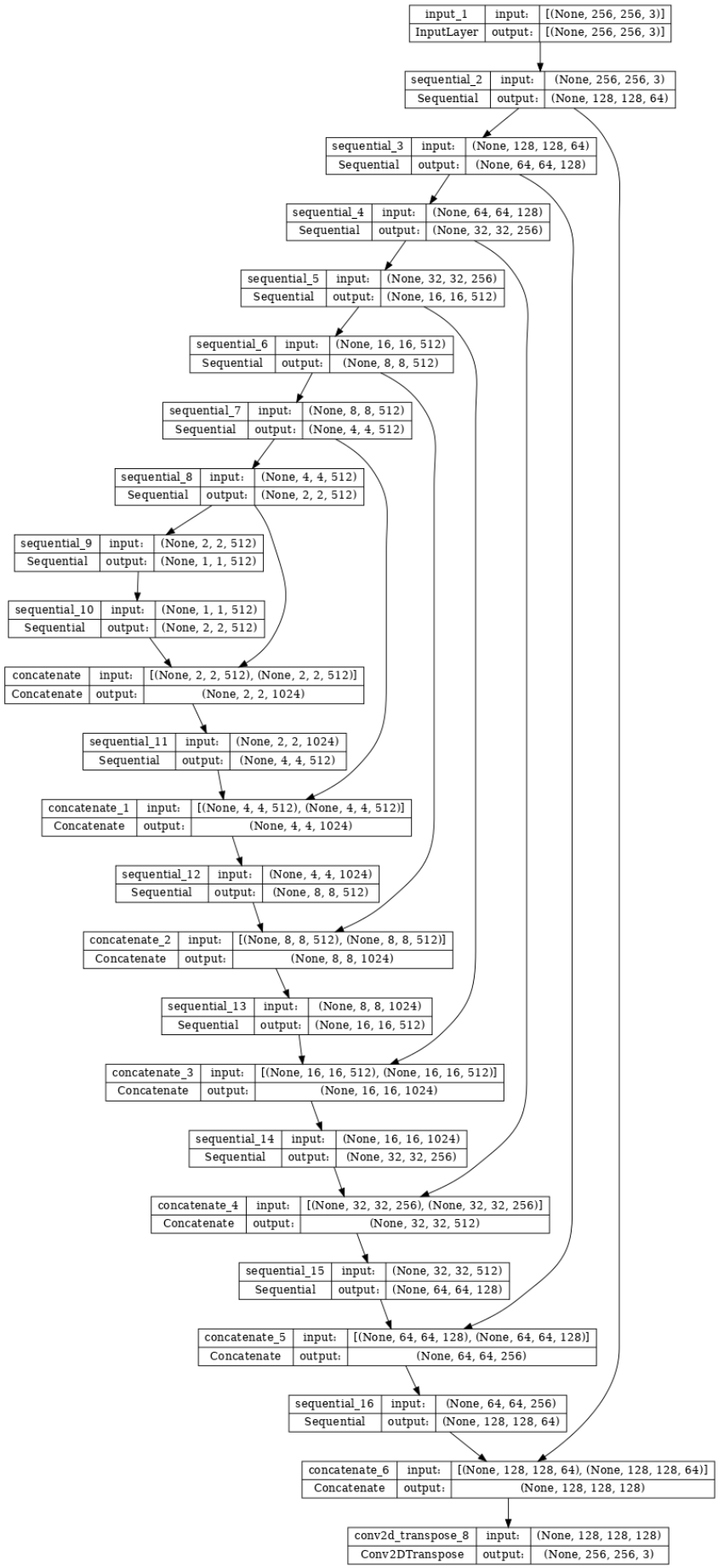


Figure 4.2: TensorFlow implementation of the U-Net generator

4.2.2 PatchGAN Discriminator

The discriminator is a convolutional neural network that classifies the real and fake images. The discriminator architecture is such that each convolutional block in the discriminator consists of a convolution layer, a batch normalization layer, and a Leaky ReLU activation layer. The PatchGAN discriminator architecture is such that it only penalizes the structure at the scale of patches. This discriminator tries to classify if each N x N patch in an image is real or fake.

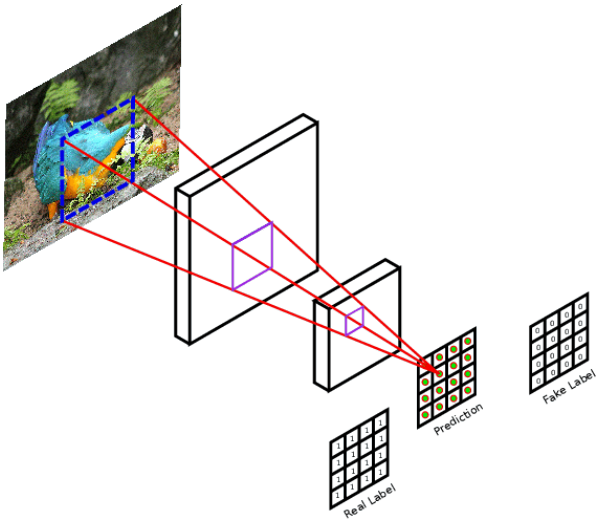


Figure 4.3: PatchGAN discriminator.

Each value of the output matrix in fig. 4.3 represents the probability of whether the corresponding image patch is real or it is artificially generated.

TensorFlow implementation

The TensorFlow implementation of the PatchGAN discriminator is composed by the following layers (see figure 4.4):

- **Input**: 2 input layers, one for the real image and one for the generated image.
- **Concatenate**: the two input layers are concatenated along the channel axis.
- **Downsampling**: the concatenated input is downsampled using 3 convolutional layers, each convolutional layer is composed by a convolutional layer, a batch normalization layer, and a Leaky ReLU activation layer.
- **Output**: the output layer is a 30×30×1 matrix, where each patch of the output classifies a 70×70 portion of the input image as real or fake.

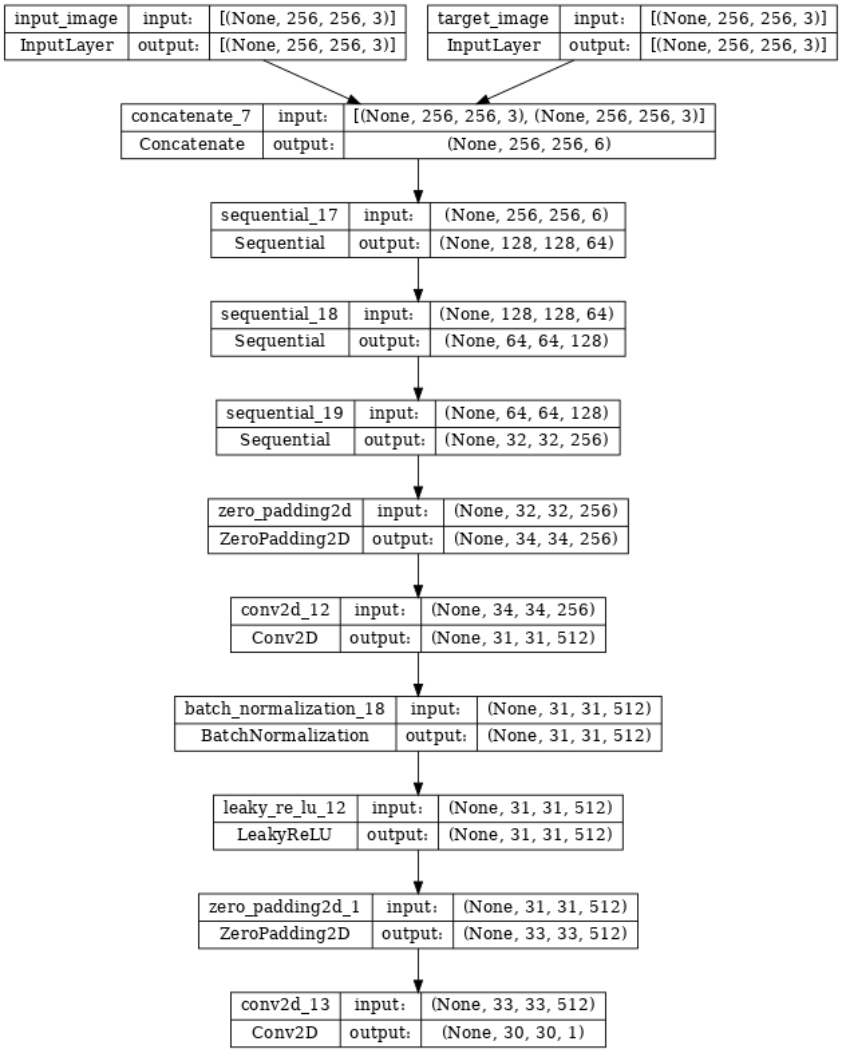


Figure 4.4: TensorFlow implementation of the PatchGAN discriminator

4.2.3 Adam optimizer

The Adam optimizer is a widely used optimization algorithm for training neural networks. It was introduced by Diederik P. Kingma and Jimmy Ba in their paper titled "Adam: A Method for Stochastic Optimization'[2] published in 2015. Adam stands for Adaptive Moment Estimation and combines the benefits of two other optimization techniques: AdaGrad and RMSProp. It maintains adaptive learning rates for each parameter, automatically adjusting the learning rate based on the gradient's past behavior. By utilizing first and second moments of the gradients, Adam updates the parameters to accelerate convergence and handle different types of neural networks effectively. It can be used instead of the classical stochastic gradient descent procedure to update network weights iterative based on training data. According to Kingma et

[2]Diederik P. Kingma and Jimmy Ba. "Adam: A Method for Stochastic Optimization". In: *International Conference on Learning Representations* (). URL: https://arxiv.org/abs/1412.6980.

al., the method is "computationally efficient, has little memory requirements, invariant to diagonal rescaling of gradients, and is well suited for problems that are large in terms of data and/or parameters".

Following the pix2pix paper, the Adam optimizer is used with a learning rate of 0.0002 and momentum of 0.5 for both the generator and the discriminator.

4.2.4 Training process

For the generator training the procedure is the following illustrated in fig. 4.5, Meanwhile for the discriminator training the procedure is the following illustrated in fig. 4.6.

The training process begins with pairs of input images and their corresponding target output images. The generator network takes the input image as input and generates a synthesized output image. The discriminator network, on the other hand, receives both the synthesized output image from the generator and the real target output image. The discriminator's objective is to correctly classify whether the input image is real or synthesized.

The training process involves alternating between two steps: generator update and discriminator update. In the generator update step, the generator parameters are updated to minimize the discrepancy between the synthesized output image and the target output image. This is typically done by minimizing a pixel-wise loss function, such as mean squared error or binary cross-entropy, which measures the difference between the synthesized and target images.

In the discriminator update step, the discriminator parameters are updated to improve its ability to discriminate between real and synthesized images. The discriminator is trained to correctly classify real images as real and synthesized images as fake. It aims to maximize its classification accuracy.

The training process continues iteratively, with the generator and discriminator networks playing a competitive game. The generator learns to generate more realistic and visually appealing output images that closely resemble the target images, while the discriminator becomes more skilled at distinguishing between real and synthesized images.

This adversarial training process creates a feedback loop where the generator tries to produce images that the discriminator cannot distinguish from real ones, and the discriminator continuously improves its ability to discriminate between real and synthesized images. This iterative training process helps the generator network learn to generate high-quality output images that are visually consistent with the target images.

The training process of the pix2pix generator and discriminator involves this iterative interplay, gradually improving the generator's ability to synthesize realistic output images and the discriminator's ability to distinguish between real and synthesized images.

4.2.5 User Interface

As a requirements of the project, a user interface was developed to allow the user to draw the input image. It was developed using the Python library Tkinter and it is composed by a window with a menu bar and a canvas where the user can draw. The menu bar consists of the following:

- **Colors**

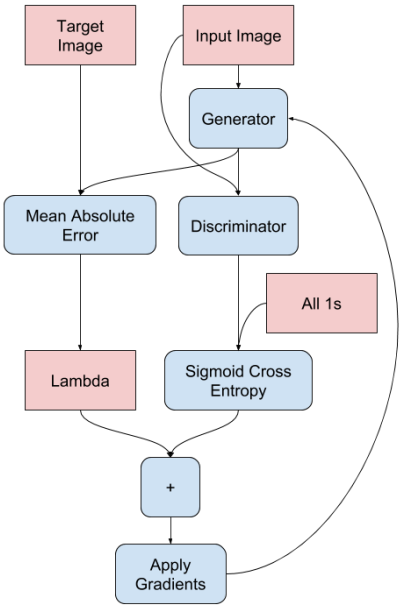


Figure 4.5: Generator training process

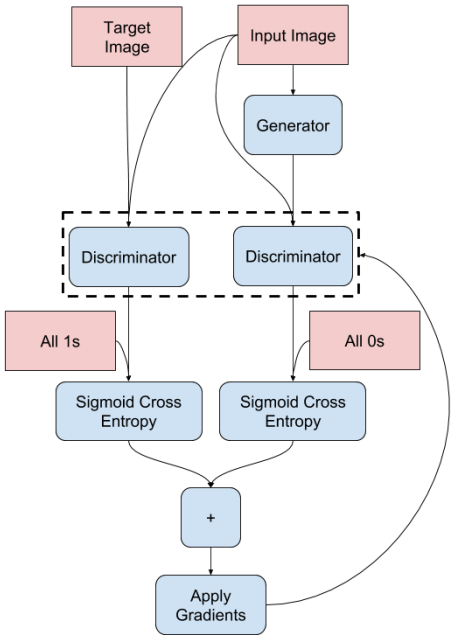


Figure 4.6: Discriminator training process

  - **Brush Color**: Set the Brush color.
  - **background Color**: Set the background color.

- **Options**
  - **Clear canvas**: Undo the last action.
  - **Generate Image**: Redo the last action.
  - **Load CAD**: Load a CAD file as input.
  - **Load PNG/JPG**: Load a .PNG file as input.
  - **Exit**: Exit the application.

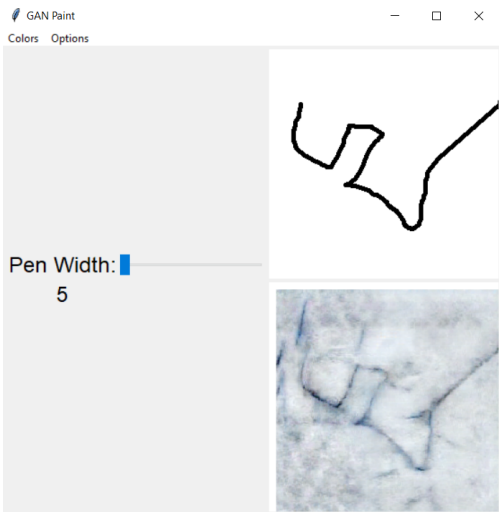


Figure 4.7: User interface

Once the input is submitted, the application will show the output in the lower canvas. And save it in the *output* folder.

Chapter 5

Verification and validation

In this chapter we will discuss the verification and validation process of the trained model. We will discuss the metrics used to evaluate the model and the results obtained.

5.1 Metrics

5.1.1 Loss function

According to the pix2pix paper[1], the loss function used is a combination of a conditional GAN loss and a L1 loss. In a general CGAN the objective function is defined as:

$$L_{cGAN}(G, D) = \mathbb{E}_{x,y}[\log D(x, y)] + \mathbb{E}_{x,z}[\log(1 - D(x, G(x, z)))] \tag{5.1}$$

here G tries to minimize this function against an adversarial D that tries to maximize it. To assess the significance of conditioning the discriminator, we also compare it to an unconditional variant where the discriminator does not have access to the input x. The loss function for this unconditional variant can be expressed as:

$$L_{cGAN}(G, D) = \mathbb{E}_{x,y}[\log D(y)] + \mathbb{E}_{x,z}[\log(1 - D(G(x, z)))] \tag{5.2}$$

Previous studies have shown the advantages of incorporating a traditional loss, such as L2 distance[2], alongside the GAN objective. While the discriminator's role remains unchanged, the generator is not only responsible for fooling the discriminator but also for producing outputs that closely resemble the ground truth in terms of L2 similarity. Additionally, we explore an alternative option by employing L1 distance instead of L2, as L1 promotes reduced blurring: The L1 loss for the generator can be defined as:

$$L_{L1}(G) = \mathbb{E}_{x,y,z}\left[\|y - G(x, z)\|_1\right] \tag{5.3}$$

Our ultimate objective is to find the optimal generator G that minimizes the loss function while simultaneously maximizing the performance of the discriminator D. This objective can be represented by the equation:

$$G^* = \arg\min_G \max_D \left(L_{cGAN}(G, D) + \lambda L_{L1}(G)\right) \tag{5.4}$$

[1] Isola et al., "Image-to-Image Translation with Conditional Adversarial Networks".

[2] D. Pathak et al. "Context Encoders: Feature Learning by Inpainting". In: *Computer Vision and Pattern Recognition* (). URL: https://arxiv.org/abs/1604.07379.

In previous conditional GAN approaches, the inclusion of Gaussian noise z alongside the input x was employed to prevent deterministic outputs and allow for the modeling of diverse distributions. However, in our experiments, this strategy was found to be ineffective as the generator learned to ignore the noise, aligning with the findings of Mathieu et al.[3]. Instead, in our final models, we introduce noise through the use of dropout applied to multiple layers of the generator during both training and testing. Despite the presence of dropout noise, we observe only minor stochasticity in the generated outputs. The development of conditional GANs that can produce highly stochastic outputs, capturing the full entropy of the conditional distributions they model, remains an open and important question for future research.

5.2 Results

5.2.1 Evaluation metrics

During the training process, the evaluation metrics were calculated every 100 epochs. For get a more accurate result, the evaluation metrics were calculated using the same couple of image-mask for each epoch, generating ten images for each epoch. Each generated image was compared with the corresponding real image.

FID score

During the training process, the FID score was calculated every 100 epochs.

Model	FID score	Epoch
M1	32,98	100
M2	26,42	200
M3	25,39	300
M4	22,29	400
M5	22,29	500
M6	22,29	600

Table 5.1: Mean FID score for each model

According to the results obtained, the best model is the **M3** model.

Inception score

During the training process, the Inception score was calculated every 100 epochs.

[3]M. Mathieu, C. Couprie, and Y. LeCun. "Deep Multi-Scale Video Prediction Beyond Mean Square Error". In: *International Conference on Learning Representations* (). URL: https://arxiv.org/abs/1511.05440.

Model	Inception score	Epoch
M1	0,6589	100
M2	0,7461	200
M3	0,7596	300
M4	0,7435	400
M5	0,7005	500
M6	0,7362	600

Table 5.2: Mean Inception score for each model

According to the results obtained, the best model is the **M3** model.

5.3 Different Network Configurations

During the training process, a problem was encountered where the image quality started to deteriorate after a certain number of epochs. Upon analyzing the loss curve, it became evident that the discriminator's loss was decreasing rapidly. This led me to believe that the discriminator was learning features at a faster rate than the generator. To address this issue, I made several adjustments to the network's hyper-parameters:

- Initially, I started training with the standard configuration and disabled discriminator training after 140k steps to prevent it from surpassing the generator's performance. Unfortunately, this modification resulted in even worse outcomes.
- I also experimented with disabling dropout on the three layers, suspecting that it might be responsible for the poor results and adversely affecting the model.
- Another attempt involved training the model for 150k steps and reducing the discriminator's learning rate to 0.0001, instead of completely disabling its training. However, this approach also yielded unsatisfactory results.

Ultimately, it was discovered that the issue did not lie with the network's hyper-parameters, but rather with the training dataset itself. The training dataset posed a challenge as it was created through an unsupervised process, resulting in the inclusion of numerous low-quality images. Some of these images contained only white spots instead of a complete line representing a vein, which affected the overall image quality. Recognizing this issue, I took steps to address it by removing such problematic images from the dataset.

As the training progressed and these flawed images were eliminated, the image quality gradually improved with each epoch. This suggests that the removal of these specific images played a crucial role in enhancing the overall performance and realism of the generated images.

Chapter 6

Conclusion

In this chapter we will present the conclusions of our work, the results obtained and the future developments of the project.

6.1 Goals achieved

In the following table we will show the goals achieved and the ones that are still in progress. (Refer to table 1.1 for the meaning of the acronyms)

Table 6.1: Reached Goals

Code	Status
M1	Achieved
M2	Achieved
M3	Achieved
D1	Achieved
D2	Not Achieved
D3	Achieved
O1	Achieved
O2	Partially Achieved

All the goals marked as achieved are fully implemented and tested. **D2** was tested but the results were not satisfactory, so it was decided to not implement it. **O2** was partially implemented, the model and the technologies are too young to be used in a manufacturing production environment.

6.1.1 CAD to GAN

The transformation from a CAD file to the final image representation of a slab involved a series of steps aimed at producing a realistic output. The objective was not only to preserve the visual characteristics and details of the CAD file's outlines but also to create an image that closely resembled a real slab.

Initially, the CAD file was converted into an image format, ensuring that the geometric information was accurately represented. This conversion facilitated further processing and manipulation of the CAD data.

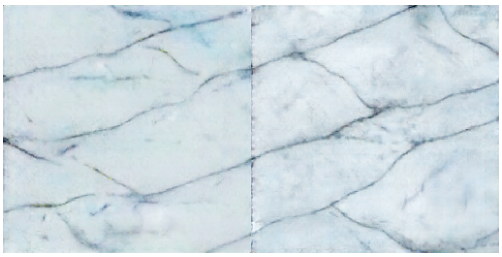


Figure 6.1: Result IMG

Following the conversion, the image file was divided into smaller portions, each conforming to the input requirements of the GAN model, with dimensions set at 256 x 256 pixels. By segmenting the image, we enabled the GAN model to process the data efficiently and generate high-quality results.

Subsequently, the GAN model was applied to each segment, generating a set of individual images. These images captured various characteristics and textures associated with slabs, enhancing the realism of the final output.

To create a cohesive representation of the slab, the individual images were carefully arranged and merged together. Through meticulous alignment and blending techniques, the resulting composite image closely resembled a genuine slab, incorporating the appearance and details that one would expect from a real-world sample.

By employing this comprehensive transformation pipeline, the CAD file was successfully translated into an input image that, when combined and processed by the GAN model, resulted in a realistic depiction of a slab. This process ensured that the final image conveyed the visual attributes and authenticity typically associated with real slabs.

Challenges

The output obtained, as depicted in Figure 6.1, reveals visible imperfections, notably the noticeable line between the two merged images. This discrepancy arises from generating the image components separately and subsequently merging them, resulting in imperfect alignment. While the main veins align reasonably well, the background color fails to exhibit proper uniformity.

During the internship period, significant efforts were dedicated to addressing this issue by enhancing the model's quality. However, the achieved results fell short of expectations. To overcome this challenge, it was proposed to generate the entire image in a single iteration. Regrettably, the current model does not support this approach, as it would necessitate increased computational resources and time, rendering it unfeasible within the constraints of the internship period.

6.2 Hourly summary

As per the initial work plan, a total of 320 hours were allocated for the project, with an intentional overestimation of hourly commitments to account for potential absences or unforeseen circumstances. The purpose of this buffer was to ensure that the project could be completed comfortably within the allocated time frame. Upon completion of the project, it was found that the actual total hours invested in the project amounted to 300 hours. This indicates that the project was completed within the expected time

frame, demonstrating a commendable consistency between the planned and actual hours spent. The adherence to the projected hour count is indicative of effective project management and efficient utilization of resources. The slight underestimation of hours compared to the initial estimate further highlights the successful management of time and effort throughout the project duration.

6.3 Acquired knowledge

During the course of the project, I had the opportunity to acquire valuable managerial and interpersonal skills while working in a collaborative research team comprising three individuals. Additionally, I delved into fields of AI that were previously unfamiliar to me, gaining comprehensive knowledge not only about AI in general but also, in particular, about Generative Adversarial Networks (GANs).

One of the significant areas of growth was in the realm of project management. Collaborating with team members allowed me to enhance my ability to coordinate tasks, set goals, and allocate resources effectively. I also developed strong communication and teamwork skills through regular interactions, discussions, and brainstorming sessions. This experience highlighted the importance of clear and concise communication, active listening, and the ability to work harmoniously towards a common objective.

In terms of technical knowledge, I embarked on an exploration of AI fields that were previously unfamiliar to me. This journey broadened my understanding of AI principles, methodologies, and techniques. More specifically, I gained in-depth knowledge about Generative Adversarial Networks (GANs) and their applications in various domains. This encompassed comprehending the underlying architecture, training procedures, and optimization techniques related to GANs.

Furthermore, I expanded my expertise in Python libraries specifically tailored for computer vision and AI applications. Through hands-on experience, I became proficient in utilizing popular libraries such as TensorFlow, PyTorch, and OpenCV. These libraries proved instrumental in implementing computer vision algorithms and working with AI models efficiently.

In summary, this project provided me with valuable insights and knowledge in both managerial and technical domains. The collaborative research environment fostered the development of essential interpersonal skills, while the exploration of unfamiliar AI fields and the utilization of Python libraries enhanced my technical proficiency. The acquired knowledge and skills from this project will undoubtedly contribute to my future endeavors in the field of AI and beyond.

6.4 Personal evaluation.

My internship experience has been truly invaluable and has provided me with a multitude of learning opportunities and personal growth. Throughout the duration of the internship, I have had the chance to work in a professional setting, applying the knowledge and skills I acquired during my studies.

One aspect of the internship that I found particularly rewarding was the exposure to real-world scenarios and challenges. This allowed me to bridge the gap between theory and practice, gaining a deeper understanding of how concepts and principles are applied in a professional context. The hands-on experience has enhanced my problem-solving abilities and critical thinking skills, enabling me to approach tasks with a more practical and solution-oriented mindset.

Moreover, the internship provided me with a chance to collaborate with a diverse team of professionals. This collaborative environment allowed me to learn from others, exchange ideas, and contribute to meaningful projects. Working alongside experienced colleagues has not only expanded my technical knowledge but has also helped me refine my interpersonal skills such as effective communication, teamwork, and adaptability.

also appreciated the mentorship and guidance provided by my supervisor throughout the internship. Their support and feedback have been instrumental in my growth and development. They provided valuable insights, challenged me to think outside the box, and encouraged me to take ownership of my work. This guidance has not only enhanced my technical skills but has also boosted my confidence in tackling complex tasks and projects.

In terms of personal growth, this internship has helped me develop a greater sense of professionalism and work ethic. It has instilled in me a strong sense of responsibility, time management, and the importance of meeting deadlines. The experience has also reinforced my passion for the field and has motivated me to continue exploring and expanding my knowledge beyond the internship.

Overall, my internship experience has been incredibly rewarding. It has provided me with practical skills, industry exposure, and personal growth opportunities. I am grateful for the chance to apply my knowledge, collaborate with professionals, and learn from experienced mentors. The lessons and experiences gained during this internship will undoubtedly have a lasting impact on my future career endeavors.

Acronyms and abbreviations

API Application Programming Interface. 42

CAD Computer-aided Design. 42

CGAN Conditional Generative Adversarial Network. 42

CNC Computerized Numerical Control. 42

CUDA Compute Unified Device Architecture. 42

CuDNN CUDA Deep Neural Network. 42

FID Fréchet Inception Distance. 42

GAN Generative Adversarial Network. 42

GPU Graphics Processing Unit. 42

HED Holistically-Nested Edge Detection. 42

IOT Internet of Things. 42

IS Inception Score. 42

ML Machine Learning. 43

POC Proof Of Concept. 43

Glossary

API *API, Application Programming Interface.* It is a set of clearly defined methods of communication between various software components. 23, 41

CAD *CAD, Computer-aided Design.* It is the use of computers to aid in the creation, modification, analysis, or optimization of a design. 3, 41

CGAN *CGAN, Conditional Generative Adversarial Network.* It is a type of GAN that uses additional information to generate images. 21, 26, 41

CNC *CNC, Computerized Numerical Control.* It is a computerized manufacturing process in which pre-programmed software and code controls the movement of production equipment. 1, 3, 41

CUDA *CUDA, Compute Unified Device Architecture.* It is a parallel computing platform and application programming interface model created by Nvidia. 22, 41

CuDNN *CuDNN, CUDA Deep Neural Network.* It is a GPU-accelerated library of primitives for deep neural networks. 23, 41

FID *FID, Fréchet Inception Distance.* It is a metric used to evaluate the quality of generated images. 9, 10, 41

GAN *GAN, Generative Adversarial Network.* It is a class of machine learning systems invented by Ian Goodfellow in 2014. Two neural networks contest with each other in a game. Given a training set, this technique learns to generate new data with the same statistics as the training set. 3–6, 9–12, 15, 24, 41

GPU *GPU, Graphics Processing Unit.* It is a specialized electronic circuit designed to rapidly manipulate and alter memory to accelerate the creation of images in a frame buffer intended for output to a display device. 22, 23, 26, 41

HED *HED, Holistically-Nested Edge Detection.* It is a state-of-the-art method for edge detection. 19, 20, 41

IOT *IOT, Internet of Things.* It is the network of physical devices, vehicles, home appliances, and other items embedded with electronics, software, sensors, actuators, and connectivity which enables these things to connect and exchange data. 1, 41

IS *IS, Inception Score.* It is a metric used to evaluate the quality of generated images. 9, 10, 41

ML *ML, Machine Learning.* It is the study of computer algorithms that improve automatically through experience. It is seen as a subset of artificial intelligence. 2, 41

POC *POC, Proof Of Concept.* It is a realization of a certain method or idea in order to demonstrate its feasibility, or a demonstration in principle with the aim of verifying that some concept or theory has practical potential. 4, 41

Bibliography

Book bibliography

James P. Womack, Daniel T. Jones. *Lean Thinking, Second Editon*. Simon & Schuster, Inc., 2010.

Wang, Xintao et al. *Real-ESRGAN: Training Real-World Blind Super-Resolution with Pure Synthetic Data*. 2021.

Web-Site bibliography

Manifesto Agile. URL: http://agilemanifesto.org/iso/it/.

Meijering Filter. URL: https://scikit-image.org/docs/stable/api/skimage.filters.html#rbd62388c4e81-1 (cit. on p. 18).

OpenCV: Canny Edge Detection. URL: https://docs.opencv.org/3.4/da/d22/tutorial_py_canny.html (cit. on p. 17).

OpenCV: Thresholding Operations. URL: https://docs.opencv.org/4.x/d7/d4d/tutorial_py_thresholding.html (cit. on p. 17).

Article bibliography

Bau, D. et al. "Network Dissection: Quantifying Interpretability of Deep Visual Representations". In: *Computer Vision and Pattern Recognition* (). URL: https://arxiv.org/abs/1704.05796 (cit. on p. 8).

Borji, Ali. "Pros and Cons of GAN Evaluation Measures". In: *arXiv preprint arXiv:1802.03446* abs/1802.03446 (). URL: https://arxiv.org/abs/1802.03446 (cit. on pp. 7, 8, 10).

Chen, X. et al. "InfoGAN: Interpretable Representation Learning by Information Maximizing Generative Adversarial Nets". In: *Advances in Neural Information Processing Systems* (). URL: https://arxiv.org/abs/1606.03657 (cit. on p. 8).

Dosovitskiy, Alexey and Thomas Brox. "Generating Images with Perceptual Similarity Metrics based on Deep Networks". In: *Advances in Neural Information Processing Systems* (). URL: https://arxiv.org/abs/1602.02644.

Goodfellow, I. et al. "Generative Adversarial Networks". In: *Advances in Neural Information Processing Systems 27* (), pp. 2672–2680. URL: http://papers.nips.cc/paper/5423-generative-adversarial-nets.pdf (cit. on p. 5).

Heusel, Martin et al. "GANs Trained by a Two Time-Scale Update Rule Converge to a Local Nash Equilibrium". In: *arXiv preprint arXiv:1706.08500* abs/1706.08500 (). URL: http://arxiv.org/abs/1706.08500 (cit. on p. 10).

Higgins, I. et al. "beta-VAE: Learning Basic Visual Concepts with a Constrained Variational Framework". In: (). URL: https://arxiv.org/abs/1711.00464 (cit. on p. 8).

Huang, Xun et al. "Stacked Generative Adversarial Networks". In: *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)* (). URL: https://arxiv.org/abs/1612.04357 (cit. on p. 7).

Isola, Phillip et al. "Image-to-Image Translation with Conditional Adversarial Networks". In: *arXiv preprint arXiv:1611.07004* abs/1611.07004 (). URL: http://arxiv.org/abs/1611.07004 (cit. on pp. 24, 34).

Kingma, Diederik P. and Jimmy Ba. "Adam: A Method for Stochastic Optimization". In: *International Conference on Learning Representations* (). URL: https://arxiv.org/abs/1412.6980 (cit. on p. 30).

Lin, Z. et al. "PacGAN: The power of two samples in generative adversarial networks". In: *arXiv preprint arXiv:1712.04086* (). URL: https://arxiv.org/abs/1712.04086 (cit. on p. 7).

Mathieu, M., C. Couprie, and Y. LeCun. "Deep Multi-Scale Video Prediction Beyond Mean Square Error". In: *International Conference on Learning Representations* (). URL: https://arxiv.org/abs/1511.05440 (cit. on p. 35).

Mathieu, M. F. et al. "Disentangling Factors of Variation in Deep Representation using Adversarial Training". In: *Advances in Neural Information Processing Systems* (). URL: https://arxiv.org/abs/1611.03383 (cit. on p. 8).

Pathak, D. et al. "Context Encoders: Feature Learning by Inpainting". In: *Computer Vision and Pattern Recognition* (). URL: https://arxiv.org/abs/1604.07379 (cit. on p. 34).

Radford, A., L. Metz, and S. Chintala. "Unsupervised Representation Learning with Deep Convolutional Generative Adversarial Networks". In: *arXiv preprint arXiv:1511.06434* (). URL: https://arxiv.org/abs/1511.06434 (cit. on p. 8).

Salimans, Tim et al. "Improved Techniques for Training GANs". In: *arXiv preprint arXiv:1606.03498* abs/1606.03498 (). URL: http://arxiv.org/abs/1606.03498 (cit. on p. 9).

Srivastava, A. et al. "Veegan: Reducing Mode Collapse in GANs using Implicit Variational Learning". In: *Advances in Neural Information Processing Systems* (). URL: https://arxiv.org/abs/1705.07761 (cit. on p. 7).

Szegedy, C. et al. "Rethinking the Inception Architecture for Computer Vision". In: *Computer Vision and Pattern Recognition* (). URL: https://arxiv.org/abs/1512.00567 (cit. on p. 10).

Xiao, C. et al. "Generating adversarial examples with adversarial networks". In: *arXiv preprint arXiv:1801.02610* (). URL: https://arxiv.org/abs/1801.02610 (cit. on p. 7).

Xie, S. and Z. Tu. "Holistically-Nested Edge Detection". In: *International Conference on Computer Vision* (). URL: https://arxiv.org/abs/1504.06375 (cit. on p. 19).

Yi, Z. et al. "DualGAN: Unsupervised Dual Learning for Image-to-Image Translation". In: *arXiv preprint arXiv:1704.02510* (). URL: https://arxiv.org/abs/1704.02510 (cit. on p. 7).

Zeiler, M. D. and R. Fergus. "Visualizing and Understanding Convolutional Networks". In: *European Conference on Computer Vision* (). URL: https://arxiv.org/abs/1311.2901 (cit. on p. 8).

Zhang, Han et al. "StackGAN: Text to Photo-realistic Image Synthesis with Stacked Generative Adversarial Networks". In: *IEEE International Conference on Computer Vision (ICCV)* (). URL: https://arxiv.org/abs/1612.03242 (cit. on p. 7).