ASTON UNIVERSITY

Automatic Learning of Augmentation Strategies Based on Deep Learning Generative Models

---

Renato Barros Arantes

Doctor of Philosophy

December, 2022

©Renato Barros Arantes, 2022

Renato Barros Arantes asserts her moral right to be identified as the author of this thesis

This copy of the thesis has been supplied on condition that anyone who consults it is understood to recognise that its copyright belongs to its author and that no quotation from the thesis and no information derived from it may be published without appropriate permission or acknowledgement.

ASTON UNIVERSITY

Automatic Learning of Augmentation Strategies Based on Deep Learning Generative Models

Renato Barros Arantes

Doctor of Philosophy

December, 2022

The limited quantity of training data can hamper supervised machine learning methods that generally need large amounts of data to avoid overfitting. Data augmentation has a long history of using machine learning algorithms and is a straightforward method to overcome overfitting and improve model generalisation. However, data augmentation schemes are typically designed by hand and demand substantial domain knowledge to create suitable data transformations. This dissertation introduces a new deep learning Generative Adversarial Network (GAN) method for image synthesis that automatically learns an augmentation strategy appropriate for sparse datasets and can be used to improve pixel-level semantic segmentation accuracy by filling the gaps in the training set. The contributions of this thesis are summarised as follows. (1) Initially, in the image synthesis domain, we propose two new generative methods based on GAN that can synthesise arbitrary-sized, high-resolution images based on a single source image. (2) Next, for the first time, by using a loss function constrained by semantic segmentation, we introduce a new GAN-based model that does label-to-image translation and delivers state-of-the-art results as an augmentation strategy. (3) Additionally, this thesis presents the first strong evidence that data density correlates with the improvement brought about by an augmentation algorithm based on GAN.

Keyword: GAN, Data augmentation, Semantic segmentation.

Acknowledgements

Words cannot express my gratitude to Dr George Vogiatzis, who was responsible for raising all the needed resources to finance my research and for whom I will forever be grateful. Without him, this dream would have never happened. Additionally, this endeavour would not have been possible without the generous support of my ex associate supervisor, Diego Faria, for his guidance, friendship and the precious time he invested in me throughout this journey. Also, my beloved friend Dr Elizabeth Wanner, who introduced me to George and always provided me with invaluable advice. Furthermore, to Maria Chli, I also must express my gratitude for always being so kind to me and being responsible for providing the funds to extend this journey one more year.

Similarly, I wish to thank my qualifying report examiners, Dr Aniko Ekart and Dr Luis Manso, who is currently my associate supervisor, for their time and helpful advice.

Finally, I would like to thank my siblings and friends for their constant support.

Publications

- [7] **ICVS 2019 - International Conference on Computer Vision Systems:** B. Arantes, Renato, George Vogiatzis and Diego Faria. *"QuiltGAN: an adversarially trained, procedural algorithm for texture generation"*. In Computer Vision Systems: 12th International Conference, ICVS 2019, Thessaloniki, Greece, September 23-25, 2019, Proceedings 12, pp. 423-432. Springer International Publishing, 2019. Contribution described in Chapter 3.

- [6] **ISVC 2020 - International Symposium on Visual Computing:** B. Arantes, Renato, George Vogiatzis and Diego Faria. *"rcGAN: Learning a Generative Model for Arbitrary Size Image Generation"*. In Advances in Visual Computing: 15th International Symposium, ISVC 2020, San Diego, CA, USA, October 57, 2020, Proceedings, Part I 15, pp. 80-94. Springer International Publishing, 2020. Contribution described in Chapter 4.

- [4] **ISVC 2020 - International Symposium on Visual Computing:** B. Arantes, Renato, George Vogiatzis and Diego Faria. *"CSC-GAN: Cycle and Semantic Consistency for Dataset Augmentation"*. In Advances in Visual Computing: 15th International Symposium, ISVC 2020, San Diego, CA, USA, October 57, 2020, Proceedings, Part I 15, pp. 170-181. Springer International Publishing, 2020. Contribution described in Chapter 5.

- [5] **IMAVIS 2022 - Image and Vision Computing:** B. Arantes, Renato, George Vogiatzis and Diego Faria. *"Learning an augmentation strategy for sparse datasets"*. Elsevier, 2022. 117, p.104338. Contribution described in Chapter 6.

Contents

List of Illustrations

Figures

Tables

List of Abbreviations

GAN	Generative Adversarial Network
DCGAN	Deep Convolutional Generative Adversarial Network
rcGAN	randomly conditioned DCGAN
CSC-GAN	Cycle and Semantic Consistent GAN
FID	Fréchet inception distance
SIFID	Single Image Fréchet inception distance
mIoU	Mean Intersection-Over-Union

Part I

Introduction and Background

Chapter 1

Introduction

Computer vision is a field of study within artificial intelligence and computer science that focuses on enabling computers to interpret and research visual information from the world around them. For example, computer vision aims to teach computers to recognise, interpret, and understand images and videos as humans do. This involves developing algorithms and techniques that enable computers to process and analyse images and videos to extract relevant information, such as objects, shapes, colours, patterns, and movements. Computer vision has numerous applications, including object recognition, facial recognition, image and video search, self-driving cars, medical imaging, and surveillance.

Deep learning is an important and powerful tool in the field of computer vision. Deep learning refers to a set of machine learning techniques based on artificial neural networks designed to mimic the structure and function of the human brain. In computer vision, deep learning algorithms can automatically learn and extract features from images and videos without manual feature engineering. This is achieved by training the neural network on a large dataset of labelled images, where the network learns to identify patterns and features that are relevant to the task at hand. Deep learning has enabled significant advances in computer vision, particularly in areas such as object recognition [78, 80, 105, 106], image classification [66, 100, 104, 115, 142], and image segmentation [18, 30, 36, 45, 48, 73, 132]. For example, deep learning models such as convolutional neural networks (CNNs) have become the state-of-the-art method for image recognition, achieving accuracy rates that rival or exceed human performance on some tasks.

However, these deep learning techniques need massive amounts of data to fulfil

their potential as powerful machine learning methods. Furthermore, to fulfil this potential, standard deep neural networks need to learn hundreds of millions of parameters, requiring many training data passes. On tiny datasets, running many iterations can result in overfitting, which is typically solved by one or more of the following: applying regularisation, acquiring more data and performing data augmentation. The latter strategy is often restricted to randomised manipulation of an existing dataset [26, 44, 66, 77, 113, 115]; in the image domain, for example, standard augmentation includes horizontally flipping the source image or translating it.

Data augmentation is an efficient way to increase the quantity and diversity of available data. Additionally it helps to teach the model about invariance in the data domain [28]. Although there is a massive effort in developing better network architectures [44, 47, 52, 104, 116, 120, 142], it has been shown [66, 93] that data augmentation can also efficiently be used to incorporate data invariance and can significantly boost the generalisation of existing deep learning models. Unfortunately, an augmentation method providing an improvement for a given dataset in a given application cannot always be successfully extended to other datasets and applications [28]. Therefore, automatically finding augmentation strategies from data is becoming a trend and is emerging as a promising strategy for planning augmentation approaches [28, 29, 74, 71, 75, 125].

In this vein, this thesis presents a GAN-based system that can be viewed as an automatically finding augmentation method for applying to the semantic segmentation problem. We show that our strategy learns an augmentation policy where the synthetic images generated by our method can be regarded as missing data points in the training set, as shown in Figure 1.1. Based on the evidence brought by our experiments, this thesis also demonstrates that data density is a factor that can hinder the improvement provided by a GAN-based augmentation algorithm.

With our automatic learning augmentation strategy, we improve the generalisation of three state-of-the-art semantic segmentation networks, *i.e.* Criss-Cross Network (CCNet) [53], DeeplabV3 [19] and Fully-Convolutional Network (FCN) [110]. This is a remarkable achievement if we consider that no new data or preliminary information has been provided to the model, especially considering that CCNet, FCN, and DeeplabV3 are already among the best architectures in extracting generalisable

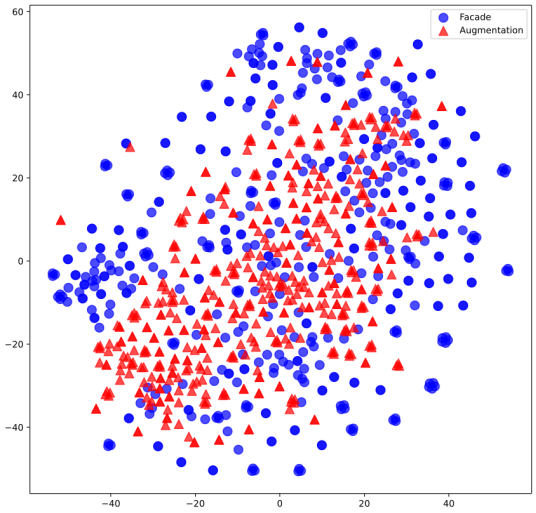


FIGURE 1.1: **Our method in action.** A t-SNE [83] plot of the extracted VGG16's features [116] on the Facades dataset [103] and the augmentation data generated by our method. Our method provides augmentation data that boost pixel-level semantic segmentation accuracy by filling the original dataset gaps.

knowledge from any data set.

1.1 Overall Aim and Research Questions

This thesis aims to contribute to the field of computer vision, in particular to the problem of data augmentation. Data augmentation is a crucial technique in deep learning. It is about increasing a training dataset's size and diversity to enhance a machine-learning model's performance. This is why we decided to focus on this problem. Data augmentation involves creating new training data examples by applying various transformations to existing data. In computer vision, it can include things like flipping, rotating, zooming, cropping, and changing the brightness and contrast of images.

One of the main advantages of data augmentation is that it can help reduce overfitting, which occurs when a deep learning model becomes too specialized for the training data and does not generalize well to new, unseen data. By increasing the diversity of the training data through data augmentation, a model can learn to recognize underlying patterns and features in the data more robustly and therefore perform better on new, unseen data.

Data augmentation is not the only solution for improving the performance of a deep learning model. Other techniques include transfer learning, where a pre-trained model is fitted to a new dataset, and regularization, which involves adding constraints to the model to prevent overfitting. However, data augmentation is often a simple and effective way to improve model performance, mainly when working with limited amounts of training data.

Data augmentation is particularly useful in image recognition tasks where there may be limited examples of certain classes of objects or variations in how images are captured or presented. It could also be helpful in other domains, such as natural language processing and audio processing, where similar techniques can be used to generate new examples from training data.

Therefore data augmentation is an important technique in deep learning that can help improve model performance by increasing the size and diversity of the training dataset. While it's not the only solution, it's often a simple and effective way to address problems like overfitting and limited training data.

Especially, this thesis addresses the following research questions:

Q1. Synthesising an arbitrary-sized, high-resolution image from a single constraint source image is important in computer vision because it can be used for various applications such as image editing, image restoration, and image generation. Hence data augmentation. Therefore, how can we synthesise an arbitrary-sized, high-resolution image from a single constraint source image?

Q2. To what extent can deep generative modelling methods such as GAN be applied to data augmentation to improve the results of semantic segmentation models?

Q3. Does the addition of semantic loss and self-attention mechanism improve the quality of our GAN-generated images, and if so, to what extent?

Q4. How does the density of the dataset impact the enhancement brought about by the GAN-generated images as an augmentation method for the per-pixel classification problem?

Research questions **Q2**, **Q3**, and **Q4** are mainly related to the data augmentation problem, and research question **Q1** is related to the image synthesis problem. As explained in the previous paragraph, image synthesis is associated with the data augmentation problem; therefore, we decided to investigate it.

1.2 Contributions

This section describes the contributions developed to address the research questions presented in the previous section. The work detailed in this thesis investigates each of these research questions and contributes to advancing the state of the art in computer vision as follows:

C1. We cover **Q1** in two steps. Initially, we investigate the texture synthesis process by proposing a method called *QuiltGAN* that generates high-resolution images based on a single constraint source image. Next, we propose a new method called *randomly conditioned GAN* or abbreviated as *rcGAN*. It also synthesises high-resolution images based on a single constraint image similar to the proposed *QuiltGAN* system. However, it is much more efficient. Our *rcGAN* algorithm can produce an almost infinite collection of variations from a single input image with sufficient variability while preserving essential large-scale constraints;

C2. To answer question **Q2**, we present two new generative models, one based on the CycleGAN [140] model and the other based on the SPADE [92] model. We demonstrate that these two models alone can deliver what can be considered a suitable machine learning augmentation strategy for semantic segmentation. However, further improvements can be added with our new semantic reinforcement loss or a self-attention [136] mechanism;

C3. To answer **Q3**, we applied several variants of the CycleGAN and SPADE models to four different datasets, three among them are public datasets and standards in the computer vision literature, and studied the influence of the synthetic

images generated by these models on the performance of three different semantic segmentation models. Using the FID [49] metric, we demonstrate that image quality improves when our models are enhanced with our semantic loss or self-attention mechanism;

To address **Q4**, we propose a density assessment metric that we use to access dataset density and relate it to semantic segmentation performance. Later, we demonstrated that data density is a factor that can affect the performance brought about when using GAN's generated images as a data augmentation method for semantic segmentation models. This is because data augmentation allows variations of the existing samples, effectively increasing the amount of data the model can learn from. When the dataset is sparse, data augmentation methods can generate new samples and improve model performance. On the other hand, if the dataset is already dense, then the effectiveness of data augmentation methods may be limited. In this case, adding more examples through data augmentation may provide little benefit.

1.3 Thesis Outline

This thesis is structured in four parts. The first part is the introduction, in which the motivations, contributions and basic techniques are detailed. The second part describes the proposed algorithm synthesising high-resolution images based on a single restriction image source. The third part contains our machine learning augmentation strategy method. Finally, in the last part, the conclusions and final considerations are presented.

Part I. Introduction and Background

CHAPTER 2 presents the technical basis and fundamental computer vision techniques applied in the following parts of the thesis, such as Adversarial Generative Network (GAN), semantic segmentation and evaluation metrics.

Part II. Image Synthesis Algorithm

In this part, we propose two deep generative models based on conditional GAN (cGAN) that synthesise images based on a single source image.

CHAPTER 3 presents the first method, named QuiltGAN, that is aimed to generate texture and is inspired by the Image Quilting algorithm proposed by Efros *et al*. [34].

CHAPTER 4 presents the second method, named rcGAN, which can be regarded as an evolution of the QuiltGAN algorithm, which is aimed to synthesise arbitrary sized images based on a single constraint source image.

Part III. Machine Learning Augmentation

In this part, we present two improvements in deep generative models based on GANs that aim to use them as a data augmentation method.

CHAPTER 5 presents our first augmentation method that is based on Cycle-GAN [140]. This method can improve the overall performance of a semantic segmentation model and is named CSC-GAN. It aims to synthesise images based on a new cycle-consistent, adversarially trained image-to-image translation with a loss function constrained by semantic segmentation.

CHAPTER 6 presents our second augmentation approach based on the SPADE model [92], connected with the self-attention adversarial network (SAGAN) [136], which enables attention-driven, long-range dependency modelling for image synthesis tasks. Moreover, it can provide further improvements compared to our previous CSC-GAN method.

Part IV. Conclusions and Final Remarks

Chapter 7 summarises the work presented in this thesis, presents the conclusions and highlights future lines of research.

Chapter 2

Background

This chapter provides an introduction to the fundamental computer vision concepts that serve as a starting point for this work.

2.1 Parametric and Non-Parametric Methods

Synthesising realistic and high-resolution images is one of the most challenging tasks in computer vision and machine learning investigation areas. It is important in computer vision because it can be used for various applications such as image editing, restoration, image generation, and, consequently, data augmentation. The methods developed so far can be grouped into parametric and non-parametric. Non-parametric methods have a long history and are typically based on examples or data-driven. This means that they exploit image pixels borrowed from existing images used as examples [11, 21, 35, 46]. Consequently, non-parametric methods often generate realistic images. Still, they need more generalisation capability as they are restricted by borrowing existing pixels and textures from the source dataset and are incapable of generating new patterns. Conversely, parametric approaches are trained to reproduce the realism of real image datasets while maintaining a level of randomness and variety that non-parametric models do not achieve. Deep learning models have shown a superior ability to generate realistic images that are different from the training dataset [16, 60].

2.2 Generative Adversarial Network (GAN)

The introduction of generative adversarial networks (GAN) by Goodfellow *et al*. [43], provides a new approach for image synthesis in computer vision. Compared with traditional supervised machine learning algorithms, generative adversarial networks employs the concept of adversarial training, in which two models are simultaneously trained: a generative model G that captures the data distribution, and a discriminator model D that estimates the probability that a sample came from the training data rather than G. To learn a generator distribution p_g, over the data x, the generator builds a mapping function from a prior noise distribution $p_z(z)$ to data space as $G(z;\theta_g)$. Moreover, the discriminator, $D(x;\theta_d)$, outputs a single scalar representing the probability that x came from the training data instead of p_g.

G and D are both trained simultaneously. The parameters θ_g for G are adjusted to minimise $log(1 - D(G(z)))$, and the parameters θ_d for D are adjusted to maximise $log(D(X))$, as if they are following the two-player min-max game with value function $V(G, D)$:

$$\min_G \max_D V(D,G) = \mathbb{E}_{p\sim p_{data}}[\log(D(x))] + \mathbb{E}_{z\sim p_z(z)}[\log(1 - D(G(z)))]. \tag{2.1}$$

GANs are a type of parametric method widely applied for image synthesis and once trained, the generator can be used to synthesise images based on a noise vector seed. When compared to blurry and low-resolution results from other deep learning methods [65, 1], GAN-based methods [101, 139, 84, 55] generate more realistic results on higher resolution and richer details.

2.3 Conditional GAN

A conditional GAN (cGAN) was first introduced by Mirza and Osindero in [87], where the authors proposed that a GAN can be extended to a conditional model if both the generator and discriminator are conditioned on some extra information **y**. This could be any auxiliary information, such as class labels or data from other

modalities. The conditioning can be performed by feeding $\mathbf{y}$ into both the discriminator D and generator G as an additional input layer. In the generator, the prior input noise z, randomly sampled from a uniform Gaussian, $p_z(z)$, and $\mathbf{y}$ are combined in a joint hidden representation. The proposed objective function is

$$\min_G \max_D V(D,G) = \mathbb{E}_{p\sim p_{data}}[\log(D(x|y))] + \mathbb{E}_{z\sim p_z(z)}[\log(1-D(G(z|y)))]. \quad (2.2)$$

In the generator the prior input noise $p_z(z)$, and y, the condition or constraint, are combined. In the discriminator x and y are presented as inputs to the discriminator. We also propose the use of conditional GAN in our rcGAN algorithm, but the objective function we are proposing is simpler, as we condition only the generator, and is expressed as

$$\min_G \max_D V(D,G) = \mathbb{E}_{p\sim p_{data}}[\log(D(x))] + \mathbb{E}_{z\sim p_z(z)}[\log(1-D(G(z|y)))]. \quad (2.3)$$

The noise vector z is needed because otherwise the generator G would produce deterministic outputs, and therefore fail to meet any distribution of the input data.

2.4 Semantic Segmentation

Currently, some of the most critical computer vision problems are image classification [123], object detection [124], and segmentation [53, 19, 110] in ascending order of difficulty. The image classification problem consists in associating a label to each object present in an image. In object detection, one more step is to try to identify the place where objects are present with the help of a bounding box. Finally, image segmentation brings the classification problem to a new level by accurately detecting the exact boundary of objects in the image. Also known as per-pixel classification, image segmentation or semantic segmentation performs its task by assigning a label to each pixel in a source image. See Figure 2.1 for an example where each object type has its own label or colour. Cars are represented in blue, roads in magenta, and so

FIGURE 2.1: **Semantic Segmentation.** Image segmentation performs its task by assigning a label to each pixel in a source image. This example is taken from the Cityscapes dataset [27], one of the datasets we use in our work.

on. We apply our automatic learning augmentation strategy to improve the generalisation performance of current state-of-the-art semantic segmentation networks, as described in Part 6.4.

2.5 Mean Intersection-Over-Union (mIoU)

The Jaccard Index, also acknowledged as the Intersection over Union (IoU), is the most common evaluation metric for segmentation, object detection, and tracking tasks. It is a statistic used to assess the similarities between two sets of samples. The measurement highlights the similarity between finite sample sets and is defined as the size of the intersection divided by the size of the union of two sample sets. The mathematical formula of the index is

$$J(A,B) = \frac{|A \cap B|}{|A \cup B|} = \frac{|A \cap B|}{|A| + |B| - |A \cap B|} \tag{2.4}$$

where $0 \leq J(A,B) \leq 1$, and if A and B are both empty, define $J(A,B) = 1$.

To calculate the mean intersection-over-union, first calculate the IoU for each semantic class and then calculate the average over the classes. The mIoU metric is used in all our experiments in part 6.4 to compare the performance of our semantic segmentation models under the influence of different augmentation methods.

2.6 Fréchet Inception Distance (FID)

Introduced by Heusel *et al*. in [49], Fréchet inception distance, or FID for short, captures the similarity of synthetic images to real images. Given the distribution $p(.)$ of the model samples and the distribution $p_w(.)$ of the samples from the real images, the FID is calculated as the difference between two Gaussians distributions, which is measured by the Fréchet distance [37], also known as Wasserstein-2 [126] distance. The Fréchet distance $d(.,.)$ between the Gaussian with mean and covariance $(\boldsymbol{m}, \boldsymbol{C})$ obtained from $p(.)$ and the Gaussian $(\boldsymbol{m}_w, \boldsymbol{C}_w)$ obtained from $p_w(.)$ is the FID metric, which is given by [33]

$$d^2((\boldsymbol{m}, \boldsymbol{C}), (\boldsymbol{m}_w, \boldsymbol{C}_w)) = \|\boldsymbol{m} - \boldsymbol{m}_w\|_2^2 + Tr(\boldsymbol{C} + \boldsymbol{C}_w - 2(\boldsymbol{C}\boldsymbol{C}_w)^{1/2}) \tag{2.5}$$

where $\|.\|_2$ is the Euclidean norm and Tr denotes the trace of a matrix.

FID is the most popular metric for measuring the feature distance between real and generated images. Therefore, we use the FID in part 6.4 in our experiments to compare the GAN-generated images with the ones from the real world.

2.7 Summary

In this chapter, we briefly present the difference between parametric and non-parametric generative models and Adversarial Generative Networks (GAN). We also present the necessary background that contributes to the implementation of our generative models based on deep learning, like conditional GAN and the FID metric used to compare our synthetic images with the ones from the real world. Then, in the following chapters, we will discuss in depth the relevant literature related to our contributions listed in Section 1.2.

Part II

Image Synthesis Algorithms

Chapter 3

Texture Synthesis

In this chapter, we address research question 1 (outline in section 1.2). We study the image synthesis process, which artificially generates images that contain some particular requested content. A long-standing goal in computer vision is to synthesise images with desired characteristics, such as the presence of specific objects, textures and other relevant semantic information. Instead of recreating an image from scratch, in recent decades, computer vision scientists have explored the idea of taking real-world samples as images and using them to synthesise new visualisations. With the advent of generative models of deep learning, the image synthesis process was taken to a new level, as it is now possible to synthesise an image based on text [102, 133, 137], label [92, 12] and other images [140, 55, 79, 24, 90]. More than that, instead of extracting patterns from the source images, deep learning methods can synthesise new patterns that are not present in the source images, but that also belong to the same data distribution. Therefore, a synthetic image generated by a deep generative model is a strong candidate for data augmentation, as it can increase the diversity of images that are seen by a model and consequently contribute to the improvement of its performance [41]. This assumption is key to our research, and we aim to find ways to generate samples that can improve the performance of Semantic Segmentation models.

This chapter proposes a deep generative model based on conditional GAN (cGAN) [87] that synthesises images based on a single source image. This method, named QuiltGAN, is aimed to generate texture and is inspired by the Image Quilting algorithm proposed by Efros *et al*. [34].

The remainder of this chapter is organised as follows: Related work on image

synthesis and texture generation is reviewed in Section 3.1, the proposed model is detailed in Section 3.2 and the experimental results are presented in Section 3.3. Finally, conclusions are summarised in Section 3.5.

3.1 Related Work

Generative Adversarial Networks (GAN) [43] described in 2 provide a new approach to image synthesis in computer vision. Compared to traditional supervised machine learning algorithms, GAN employs the concept of adversary training, in which two models are trained simultaneously: a generative model, G, which generates images from a data distribution, and a discriminative model D, which tries to determine if a sample came from the training data, rather than G.

Deep Convolutional Generative Adversarial Network (DCGAN) [101], have a specific style of architecture, bridging the gap between the achievement of CNNs for supervised learning and unsupervised learning. The DCGAN architecture was widely adopted [42], including our method described in this chapter, where we join the DCGAN architecture and the conditional GAN (cGAN) [87] approach. A conditional GAN (cGAN) was first introduced by Mirza *et al*. [87], where the authors proposed that a GAN can be extended to a conditional model if both the generator and discriminator are conditioned on some extra information $\mathbf{y}$. This could be any auxiliary information, such as class labels or data from other modalities. The conditioning can be performed by feeding $\mathbf{y}$ into both the discriminator and generator as an additional input layer. In the generator, the prior input noise z, randomly sampled from a uniform Gaussian, $p_z(z)$, and $\mathbf{y}$ are combined in a joint hidden representation.

Early work on texture and image synthesis related to the method presented here is proposed by Iizuka *et al*. [54]. The authors present a triple network architecture for inpainting: a completion network, a global context discriminator and a local context discriminator. The completion network is fully convolutional and used to complete the image, while both the global and the local context discriminators are auxiliary networks used exclusively for training. These discriminators are used to determine whether or not an image is consistent. The global discriminator takes the full image

as input to recognise global consistency of the scene, while the local discriminator looks only at a small region around the completed area in order to judge the quality of more detailed appearance. The method proposed by Iizuka *et al*. [54] can be used to complete an extensive variety of scenes and also can generate fragments that do not appear elsewhere in the image. Our method can also be regarded as an inpainting method because it uses conditional GANs to generate blocks of images complementary to a given model. However, differently from most of the proposed methods listed here, we aim to create an entirely new image from scratch based on a given model. This is the novelty of our proposed approach: instead of generating a patch that fills a hole in an image, the proposed method uses that image as a model to synthesise an entirely new one that is highly related to the model.

An extension of the work proposed by Pathak *et al*. [94] appeared in Iizuka [54], where the authors proposed a Context Encoder as a convolutional neural network trained to generate the contents of an arbitrary image region conditioned on its surroundings. They found that a context learns a representation that captures not just appearance but also the semantics of visual structures. Their approach is to use an alternate formulation, by conditioning only the generator (not the discriminator) on context. They also found results improved when the generator was not conditioned on a noise vector. Our method also uses that alternate formulation as it only contextualises the generator. Together with this study, our method emphasises the importance of contextualisation to capture the semantics of the visual structures in the constraint image.

PatchGAN was introduced by Isola *et al*. [55] where a simple framework can be adapted to various image generation problems. Instead of grading the whole image, a PatchGAN slides a window over the input and produces a score that indicates whether the patch is real or fake. We also use this sliding window approach to generate the dataset, which is composed exclusively of patches extracted from the constraint image. Therefore we can create a dataset based on a single image which is critical for training robust machine learning models.

Similar to the work proposed by Iizuka *et al*. [54], where a triple network architecture is used, Demir *et al*. [31] introduce a generative CNN model and a training procedure for the arbitrary hole-filling problem. The generator network takes

the corrupted image and tries to reconstruct the repaired image. They utilised the ResNet [47] architecture as the generator model with a few alterations. The critical point of their work is that they propose to design a novel discriminator network that combines a global GAN (G-GAN) structure with PatchGAN approach which they call PGGAN. The authors report considerable achievements in both visual and quantitative evaluations.

Regular GANs usually map a noise input vector to an image. Instead, Spatial GANs (SGAN) [58] extend the input noise distribution space from a single vector to a whole spatial tensor. Jetchev *et al*. lately extended SGAN to Periodic Spatial GAN (PSGAN) [13], which extend the structure of the input noise distribution by constructing tensors with different types of dimensions. In PSGAN, the input spatial tensor is composed of three parts: a local independent, a spatially global, and a periodic part. Each piece has the same spatial dimensions but can vary in their channel dimensions. PSGAN can synthesise periodic and high-resolution textures and both SGAN and PSGAN architecture, as the proposed method in this Chapter, are based on DCGAN [101]. Our method will, similarly to PSGAN, can also be trained using just a single image.

The GAN architecture has brought extra strength to the subject of synthesis and texture transfer [31, 39, 54, 55, 131, 38]. However, Gatys *et al*. introduced a new parametric texture model based only on a convolutional neural network where textures are represented by the correlations between feature maps in several layers of the network [39]. Portilla and Simoncelli describe an algorithm for synthesising random images, subject to a statistical model for texture images [98]. These are parameterised by a set of statistics constraints regarding spatial locations, orientations, and scales. Our method will, similarly to both [39] and [98] generate new textures based on a given image using a neural network. The difference in our work is that we use a patch based approach that allows our model to encode both high-frequency textures as well as large scale interactions. We can therefore synthesise larger images with coherent features *e.g*. the aligned windows in the architectural facades.

More recently, in TextureGAN [131], the authors developed a local texture loss to train a generative network that learns to synthesise objects consistent with texture suggestions and allow non-experts to create realistic visual content through a

drag and drop texture interface where users set specific textures onto sketched object boundaries. Next, using a Progressive Growing of GANs (ProGAN) [62] to synthesise images, the authors in [38] introduce TileGAN to synthesise a large-scale output, and as in our method, they present an algorithm for combining the outputs of GANs to produce a large-scale texture map without boundary artefacts. Finally, the subject of texture generation is far from exhausted, and the advent of GANs is a tool that researchers are exploring to take this subject even further, as recent papers can testify [97, 88, 138].

3.2 Methodology

(A) Source image. (B) Synthesised images.

FIGURE 3.1: **Our algorithm in action.** Source image used to train our GAN and four different synthesised outputs.

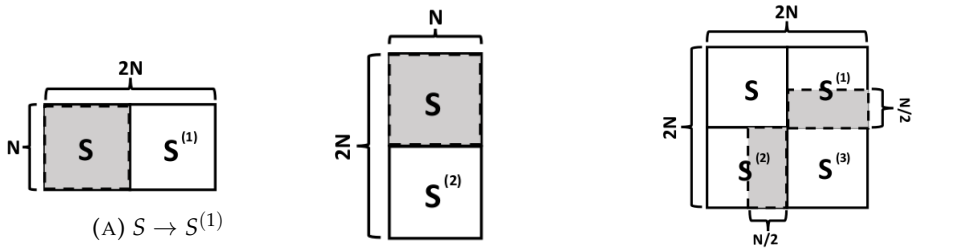
This chapter investigates a generative method that synthesises high-resolution images based on a single constraint source image. Our approach consists of three types of conditioned Deep Convolutional Generative Adversarial Networks (cDCGAN) trained to generate samples of an image patch conditional on the surrounding image regions. The cDCGAN discriminator evaluates the realism of the generated sample concatenated with the surrounding pixels that were conditioned on. This encourages the cDCGAN generator to create image patches that seamlessly blend with their surroundings while maintaining the randomisation of the standard GAN process. After training, the cDCGANs recursively generate a sequence of samples

which are then stitched together to synthesise a larger image. As a result, our algorithm is able to produce a collection of variations of a single input image that have enough variability while preserving the essential large-scale constraints. The proposed QuiltGAN method is compared with the results obtained from the *image quilting* (IQ) algorithm [34] in several different variants, refer to Section 3.3 for the results. We demonstrate that our QuiltGAN method can be superior to the simple image quilting algorithm because a GAN can generate new patches that do not exist in the constraint, while the IQ algorithm can only repeat patches from the source image.

3.2.1 Problem Formulation

Using just one image, we would like to train a GAN and obtain as a result a synthesised image that is different from the source image used to train the GAN, but at the same time shares its main characteristics. For example, we used the facade [103] in Figure 3.1a to train our GANs and we would like, at the end of the process pipeline, to obtain an image like the one in the Figure 3.1b. The Figure 3.1b was created using our proposed method, as described in 3.2.2.

The problem we are considering can be stated as a constrained image generation. Given a $N \times N$ source image S we would like to generate three blocks of size $N \times N$, named $S^{(1)}$, $S^{(2)}$ and $S^{(3)}$, in a way that they are compatible with each other. Figure 3.2 show how these blocks are conditioned on their generation.



(A) $S \rightarrow S^{(1)}$ (B) $S \rightarrow S^{(2)}$ (C) $S^{(1)}, S^{(2)} \rightarrow S^{(3)}$

FIGURE 3.2: **Conditional patches.** The three types of conditional GAN generation are shown here. The image regions that are used to condition the image are shown in gray shade. $S^{(1)}$, $S^{(2)}$ and $S^{(3)}$ are the generated patches that should be *compatible* with the condition images.

By compatible, we mean an image that can be placed alongside another image and

the edges will match and blend perfectly. We need to guarantee semantic compatibility between both images and a continuous transition between them to achieve this. Figure 3.3 is an example of a compatible synthetic image generation.

(A) The condition (B) The synthetic image

(C) The final result

FIGURE 3.3: **Continuous transition.** The condition patch (a) is used to generate a compatible patch (b), which is stitched together with it (c).

3.2.2 Proposed Method

We propose a new framework to generate constrained images based on deep convolutional generative adversarial networks (DCGAN) [101] and conditional GAN (cGAN) [87], which we named QuiltGAN [1]. Our approach is designed to work in situations where computational resources are limited, as we have access to only one GPU with 8 gigabytes of RAM. Therefore we propose three cDCGANs, that, when working together, can generate arbitrary size, high-resolution images, despite the limited computational resources. These three cDCGANs, share the same structure and are based on the same principles, which consist of a conditioned generator G and a non-conditioned discriminator D. The generator receives a random vector z concatenated with an $N \times N$ input image S as input. Its output is another $N \times N$ image S^*. Three cDCGANs were trained, as follows:

[1] `https://github.com/renaaran/fourc`

1. A pairwise cDCGAN, that given the conditioning image S generates $S^{(1)}$, that is compatible to S on the right.

2. A pairwise cDCGAN, that given the conditioning image S generates $S^{(2)}$, that is compatible to S on the bottom.

3. A cDCGAN that receives $S^{(1)}$ and $S^{(2)}$, extracts $S^{(12)}$ as a condition and generates $S^{(3)}$, that is compatible to $S^{(1)}$ above and is compatible to $S^{(2)}$ on the left.

In the process of training that output image S^* will be concatenated with S, as in Figure 3.4, to create a resulting image. The discriminator network D takes both generated images and the real image, see Figure 3.5, and strives to distinguish them while the generator network G makes an effort to fool it. Note that our method differs from [87] as we are conditioning only the generator G, letting the discriminator D as proposed by [101], without any constraint. The reason is that conditioning only the generator can increase performance because it allows the generator to learn a more direct mapping between the additional information and the generated samples while still allowing the discriminator to learn to distinguish between real and fake images based on their visual appearance. This can result in higher-quality generated samples that are better aligned with the conditioning information.

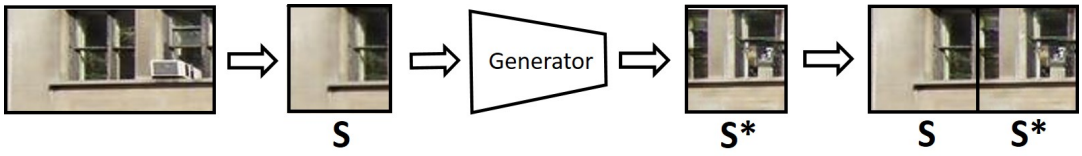


FIGURE 3.4: **Generator pipeline.** An input $N \times 2N$ image patch is split into two $N \times N$ patches, one of which (S) is used to condition the generator. The $N \times N$ output of the generator S^* is concatenated with S before it's fed to the discriminator.

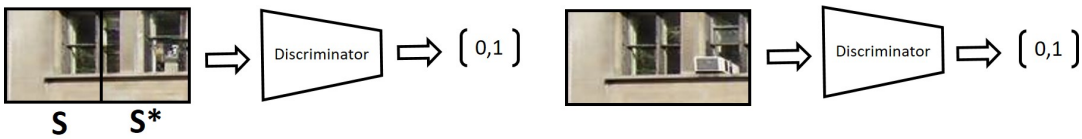


FIGURE 3.5: **Discriminator pipeline.** The discriminator is fed a series of $N \times 2N$ patches and tries to distinguishes between those extracted from actual images and those where the right $N \times N$ sub-patch is synthesised by the generator.

We condition the generator by concatenating the constraint image S with the input for the layer L_i, where i is the number of the current layer L of the generator G. For the first layer of the generator G, $i = 1$, this means concatenate the constraint image with the input noise vector sampled from a uniform Gaussian. For the subsequent layers, $i = \{2, .., n\}$, we concatenate the output of the layer L_{i-1} with the constraint image S and gives this concatenation as the input for the layer L_i.

In a traditional GAN architecture, the generator takes a noise vector as input and produces an image as output. However, the generator has no control over what type of image it generates. By conditioning each generator layer, we aim to guide the generator to produce images that perfectly match the condition and input image S. See Figure 3.6 for the overall generator network structure.

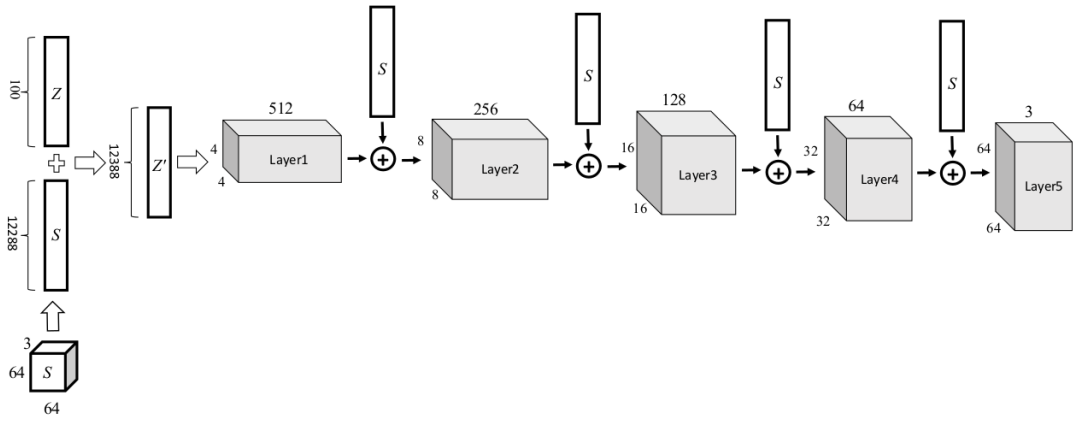


FIGURE 3.6: **Conditional GAN architecture.** First, the $64 \times 64 \times 3$ input image is reshaped into an array and then concatenated with input noise vector z. Subsequently, the output of one layer is concatenated with the condition vector S before it is fed to the next layer. Our experiments demonstrated that conditioning each layer of a Deep Convolutional Generative Adversarial Network (DCGAN) generator is a good architecture choice because it allows for greater control over the generated images.

3.2.3 Adversarial Loss

Our proposal is to extend the generative adversarial network model to a conditional model, where only the generator G is conditioned on an constraint image S. Our work differs from [87] where they condition both the generator G and the discriminator D. In that work the proposed objective function is

$$\min_G \max_D V(D,G) = \mathbb{E}_{p \sim p_{data}}[\log(D(x|y))] + \mathbb{E}_{z \sim p_z(z)}[\log(1 - D(G(z|y)))]. \quad (3.1)$$

In the generator, the prior input noise $p_z(z)$, and y, the condition or constraint, are combined. In the discriminator, x and y are presented as inputs to the discriminator. The objective of our cDCGAN is less complicated and is expressed as

$$\min_G \max_D V(D,G) = \mathbb{E}_{p\sim p_{data}}[\log(D(x))] + \mathbb{E}_{z\sim p_z(z)}[\log(1 - D(G(z|S)))]. \quad (3.2)$$

where x is the ground truth, S is the constraint image and z is the noise vector. G tries to minimise this loss function against an adversarial D that seeks to maximise it. The noise vector z is needed because otherwise the generator G would produce deterministic outputs.

3.2.4 Image Quilting

Our method is inspired by and compared to the *Image Quilting* algorithm, also known as texture synthesis, created by Alexei Efros and William Freeman in 2001 [34]. The algorithm generates large textures or images by stitching together small overlapping patches from a sample texture. The algorithm divides the sample texture into small overlapping square patches. It then randomly selects one of these patches as the starting point for the output image. The algorithm then looks for the best match for the next patch by comparing the similarity of the overlapping regions between the current patch and the available patches. Finally, the best match is selected and placed in the output image, overlapping with the previous patch.

Here is a brief description of the *Image Quilting* algorithm:

Given a set B of $N \times N$ pixels, extracted from a source image, the algorithm below will be executed to generate as output an array of blocks, whose size can vary depending on the experiment, selected from B:

1. Select at random one block from B and copy it into the top left block of the new synthesised image.

2. Go through the image to be synthesised, from left to right and from top to bottom, in steps of one block.

3. For every new location, search the set B for a block that minimises the overlap constraints, above and left. The overlap constraint is the $L2$ error norm computed on pixel values of the overlap edge. The overlap edge is $1/6$ of the size of the block. The block size is the only parameter controlled by the user and depends on the properties of a given texture: The block must be large enough to capture the relevant structures in the texture, but small enough that the interaction between these structures is left to the algorithm. Also the choice of the width of the overlap region is essential, as a narrow overlap region can result in visible seams between patches. In contrast, a wide overlap region can lead to slow and inefficient synthesis. Efros and Freeman found that a width of $1/6$ of the patch size provided a good balance between seamlessness and efficiency.

4. Compute the error surface between the newly chosen and old blocks at the overlap edge. Find the minimum cost path along this surface and make that the boundary of the new block. Paste the block.

5. Repeat until the new image is completely synthesised.

The minimal cost path through the overlaping area is computed as follow: If B_1 and B_2 are two blocks that overlap along their vertical edge, see Figure 3.7, with the regions of overlap B_1^{ov} and B_2^{ov}, respectively, then the error surface is defined as $e = (B_1^{ov} - B_2^{ov})^2$. To find the minimal vertical cut through this surface we traverse e $(i = 2..N)$ and compute the cumulative minimum error E for all paths:

$$E_{i,j} = e_{i,j} + \min(E_{i-1,j-1}, E_{i-1,j}, E_{i,-1,j+1}). \tag{3.3}$$

The minimum value of the last row in E shows the end of the minimal vertical path through the overlapping area, and one can trace back and find the path of the best cut.

3.3 Experimental Results

This section describes the experiments performed with our QuiltGAN method as proposed in 3.2.2. Here, we compare our QuiltGAN algorithm with two methods:

(A) Neighbour blocks are constrained by overlap. (B) Minimum error boundary cut.

FIGURE 3.7: **Minimum cost path.** We want to cut two overlapping blocks on the pixels where the two overlapping areas match best, where the overlap error is low.

1. The Image Quilting (IQ) algorithm, as described in 3.2.4;

2. In addition, the experiment setup section below describes an approach that mixes the IQ algorithm, and GAN generated patches.

With these experiments, we intend to compare the performance and the flexibility of QuiltGAN and IQ algorithms.

3.3.1 Experimental Setup

We propose three experiments. A baseline experiment consisting of the Image Quilting (IQ) algorithm that takes patches extracted from an image as input. A second IQ experiment, but now as input, the IQ algorithm has patches generated by the QuiltGAN method. And a third experiment that has images generated exclusively by our QuiltGAN framework.

In all experiments performed with the IQ algorithm, the block size was 64×64 pixels, the size of the DCGAN input and the generated image. The overlapping border width was $\lfloor 1/6 \rfloor$ the block size, in our case $\lfloor 64/6 \rfloor = 10$. The error was calculated using the standard $L2$ in pixel values, and the generated images are of 5×5 blocks.

The second experiment approach that mixes the IQ algorithm and the GAN-generated patches works as follows: Initially, a B set with five thousand patches of 64×64 pixels each was generated using an unconditional GAN, trained in the source image. Finally, this set B is then used to synthesise an image using the *Image Quilting* algorithm, as described in 3.2.4. The number five thousand was selected experimentally, striving to give the IQ many options for finding a good match for a particular block.

3.3.2 Evaluation

Our method was tested qualitatively and quantitatively on three images: urban extract from the facade [103] dataset, texture and artistic from the Web. Qualitative examples of our generated samples are shown in Figure 3.8. First, column two in Figure 3.8 was created using the simple version of the IQ algorithm as described in 3.2.4. Next, column three was created with the mixed approach, *i.e*. the IQ algorithm and GAN generated patches. Finally, column four was created using our proposed QuiltGAN method as described in 3.2.2.

We observe that the IQ-generated images are coherent but lack variability because they locally repeat patches from the source image. The variant of IQ that uses the GAN for generating a pool of candidate patches does better in terms of variability but lacks coherence. This is because there is nothing to force the IQ algorithm to respect longer-range pixel interactions, such as, for example, the architectural constraint of placing the building windows at regular intervals. Our QuiltGAN generated images exhibit the best of both worlds. They do provide randomisation and variability (*e.g*. none of the building windows is an exact copy of the source image (see Fig. 3.1b) while also respecting the global image constraints. The algorithm proves capable of also handling non-repeating images such as the Picasso painting (Fig. 3.8, third row). The synthesised painting respects the overall structure of the source image while providing a degree of random variation.

One failure case of our method results in a type of *fading* condition. If one cDCGAN generated patch has some errors, *i.e*. some non-realistic appearance, then the next patch generated, conditioned on it, will further deteriorate. This will result in a chain of progressively worse image patches leading to an unrealistic image. See Figure 4.11 for example. This is an artefact due to the recursive application of the cDCGAN that follows a linear path through the image.

3.4 Quantitative Evaluation

In order to assess the realism of our synthetic images and their ability to accurately represent the internal statistics of the training image, we utilised the Single-Image

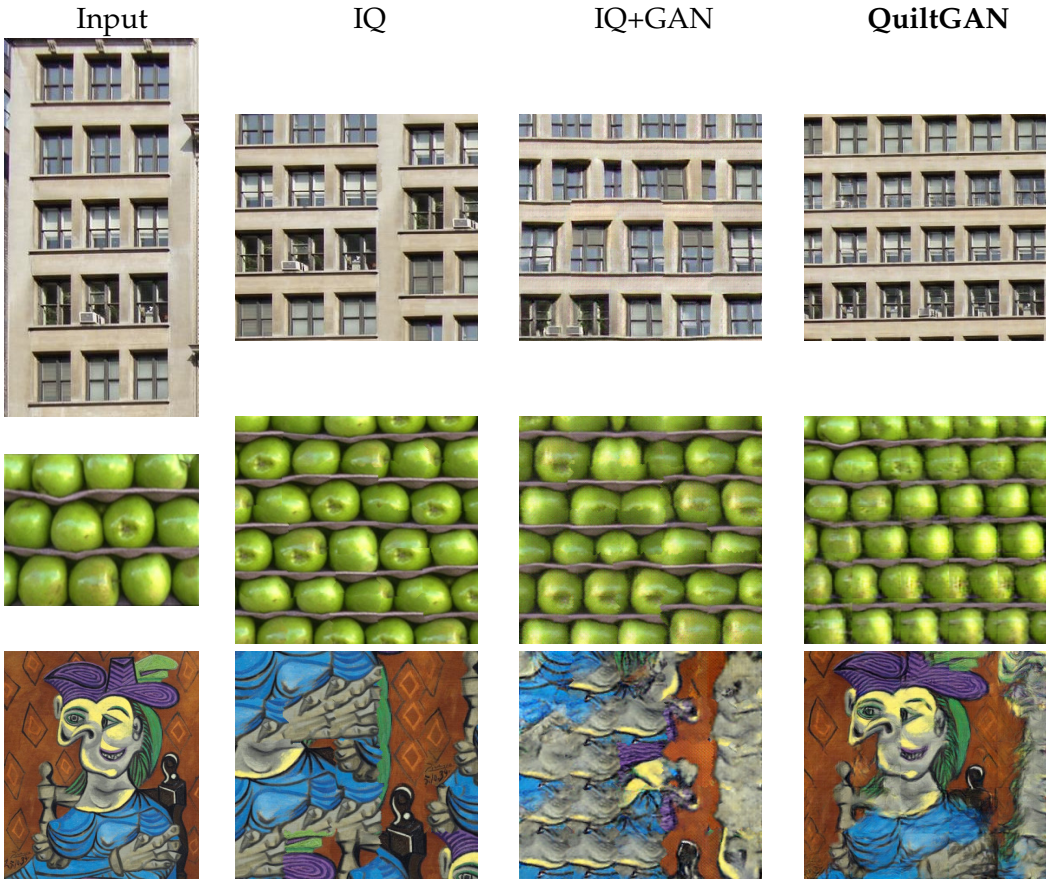


FIGURE 3.8: **Qualitative results.** Column one is the source image. Column two is generated with the Image Quilting (IQ) algorithm. Column three is IQ over GAN generated patches, and column four is our QuiltGAN system. Our method generates images that provide randomisation and variability. For example, none of the building windows in the first row is an exact copy of the source image, although it also respects global image constraints. All synthetic images are 280×280

FIGURE 3.9: **A typical failure mode.** The QuiltGAN algorithm is applied recursively and linearly through the image, occasionally causing errors to accumulate.

Version of the Fréchet Inception Distance metric, which was originally developed by the authors of SinGAN [107].

Fréchet Inception Distance (FID) [49] serves as a commonly used metric for evaluating GANs. It quantifies the disparity between the distribution of deep features in generated images and that of real images. As in our scenario, SinGAN operates with only one training image. Hence their proposal of a single image metric based on FID. Named Single Image FID (SIFID), this metric focuses on the internal training image patch statistics, rather than extracting information from a whole dataset. Instead of utilising the activation vector following the final pooling layer in the Inception Network [121], SIFID employs the internal distribution of deep features at the output of the convolutional layer just before the second pooling layer. Thus, SIFID calculates the FID between the statistics of these features in the real image and the generated sample.

The results of our QuiltGAN approach are displayed in Table 3.1. We noticed that all the images closely resembled the original patch distribution, except for the apple image. We believe this is due to an artifact that causes a lack of smooth transition between patches. You can see the artifact in Figure 3.10, especially when the image is zoomed in.

Input	**IQ**	**IQ+GAN**	**QuiltGAN**
Facade	0.967	0.406	**0.379**
Apple	**0.196**	0.362	0.213
Picasso	0.359	0.379	**0.139**

TABLE 3.1: **Experimental results**. The Single Image FID (SIFID) score evaluates each method's performance. A lower score indicates better performance.

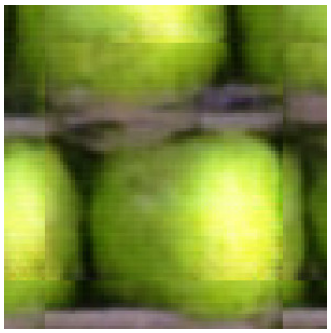


FIGURE 3.10: **Image Artifact.** The patches' transition is not smooth, resulting in a synthetic image that is out-of-the-distribution and negatively affects the SIFID score.

3.5 Summary

In this chapter, we presented our QuiltGAN method to synthesise images based on conditioned DCGANs. Our approach uses just a single image to train three GANs for generating image patches that can be stitched together to manufacture a larger image. The GANs are conditioned on an adjacent part of the image and learn how to complete it by synthesising a *matching* image region. We compare the results with the *image quilting* (IQ) algorithm [34] and conclude that the proposed method, although more complex than the IQ algorithm, can produce better results and, above all, can produce patterns that are not in the source image but are constrained by it.

Although the QuiltGAN method succeeds in generating realistic images and textures, it is expensive and challenging to train as three DCGANs need to be trained, one for each conditional patch, as explained in section 3.2.1. This can be error-prone as the training process and the results of three DCGANs must be coordinated to generate a synthetic image. Moreover, training each GAN takes over 9 hours on a single GPU NVIDIA GeForce GTX 1070 with 8 gigabytes of RAM. Consequently, each experiment run takes at least 27 hours to complete since the GANs cant be trained in parallel due to resource constraints.

Aiming to keep the process simple, less expensive for training and less suitable for error, in Chapter 4, we propose an improvement to the QuiltGAN method that merges all three GANs in a single one, offering a neat methodology also capable of synthesising an arbitrary-sized, high-resolution image from a single constraint source image.

Chapter 4

Arbitrary Size Image Generation

When rendering photo-realistic images using computer graphics, real objects and their properties such as geometry, materials or light must be explicitly modelled, normally a time consuming and expensive process. In recent years, data-driven photo-realistic image synthesis has emerged as a viable alternative for certain scene classes. Available methods can be grouped into parametric and non-parametric. Non-parametric methods have a long history and are typically based on compositing from pre-existing exemplar images [11, 21, 35, 46]. While they often succeed in generating very realistic images, they suffer from a lack of variability in the generated content. On the other hand, parametric methods based on deep neural architectures have delivered promising results in the last years. Parametric approaches are trained to reproduce the realism of real image datasets while maintaining a level of randomness and variety that non-parametric models do not achieve [16, 60]. However, with a few exceptions (e.g [107, 63]), parametric models are rarely able to scale up to full-resolution images.

In this chapter, we continue addressing research question 1 (outline in section 1.2) and investigate whether GANs [43], one of the most successful parametric models, might be employed to generate new, high-resolution images based on an individual reference image. Here we introduce *rcGAN* [1], our generative method that is capable of synthesising arbitrary sized, high-resolution images derived from a single reference image used to train our model. Our two-steps method uses a randomly conditioned Generative Adversarial Network (rcGAN) trained on patches obtained from a reference image. It can capture the reference image internal patches distribution

[1] `https://github.com/renaaran/rcGAN`

and then produce high-quality samples that share the same visual attributes with this image. After training, the *rcGAN* generates recursively an arbitrary number of samples that are then stitched together to produce an image whose size is determined by the number of samples used to synthesise it. Thus, our proposed method can provide a practically infinite number of variations of a single input image that offers enough variability while preserving the essential large scale constraints.

To further establish the resilience of our two-steps method, we experiment on many types of models, including textures, building facades, natural landscapes and compare with other strategies.

The remainder of this chapter is organised as follows: The current literature review on adversarial methods for image synthesis is presented in Section 4.1, the proposed methodology is detailed in Section 4.2, and the experimental results are presented in Section 4.3. Finally, conclusions are summarised in Section 4.5.

4.1 Related Work

In [101] Radford *et al*. introduced the deep convolutional generative adversarial networks (DCGANs) that creates a link among the achievements of CNNs for supervised learning and unsupervised learning. This model was highly influential [42], and the majority of GANs today are generally based on the DCGAN architecture, including the two-steps approach that is presented in this paper, where we combine the DCGAN architecture and the conditional GAN (cGAN) approach. Introduced by Mirza and Osindero in [87], a conditional GAN (cGAN) is a model where both the generator and the discriminator are conditioned on any information $\mathbf{y}$. This information could be an image, like our two-step method, or any additional information, such as a class label. In [87] the conditioning is done by supplying the condition $\mathbf{y}$ into both the generator and discriminator. In our method, on the other hand, we are using an alternate approach where we are conditioning only the generator as in [94].

In [54] Iizuka *et al*. introduce a triple network architecture for inpainting. A fully convolutional completion network is used to complete the image, a global context discriminator and a local context discriminator. Both are secondary networks used only during the training process. These discriminators are used to discover if an

image is coherent or not. The global discriminator gets the whole picture as input and evaluates the global coherence of the scene. In opposition, the local discriminator tries to estimate the quality of a more specific region by looking only at a small part around a painted area. The authors declare that their method can be used for inpainting a broad range of scenes and creating pieces that cannot be seen in other parts of the image. Our procedure can also be considered as an inpainting method as we use a conditional GAN to generate image blocks that match a given patch extracted from the source image, used to train our model. In the opposite direction of most works presented here, we propose to create a new image entirely from scratch, which is based on the given reference image. This approach is the originality of our proposed method: instead of generating a patch that fills a hole in an image, our prosed two-step method uses that image as a model to synthesise randomly conditioned patches that later on are stitched together to create an arbitrary sized image. The number of stitched patches determines the size of the synthetic image, each measuring 64×64 pixels.

Further development of the work introduced by Pathak *et al*. in [94] appeared in [54]. In [54], the authors introduced a Context Encoder as a CNN trained to create the contents of an arbitrary image area conditioned on its neighbourhood pixels. Their strategy is to use an alternative formulation by conditioning only the generator on the context. Our method, as said before, also uses this alternative formulation as we are conditioning only the generator. The authors also mentioned improved results when the generator was not conditioned on a noise vector.

PatchGAN was proposed by Isola *et al*. in [55] where the same method is adjusted to many image generation problems, like Pix2Pix [56] and CycleGAN [140]. Instead of evaluating the whole image, they developed a discriminator architecture that penalises structures at the scale of patches. We use the PatchGAN approach in our model as empirical experiments showed that it could slightly improve the overall quality of the generated patch.

In [31] Demir *et al*. also proposed a triple network architecture, as in [54], called PGGAN. A damaged image is the generator input which tries to restore it. The generator is based on the ResNet [47] architecture with some few modifications. For the discriminators, they join a global GAN (G-GAN), that measures the quality of the

image as a whole, and a PatchGAN that evaluates the consistency in local details.

Single image GANs. In Spatial-GAN (SGAN), Jetchev *et al*. [58] extended the input noise distribution space from a single vector to a spatial tensor and in [13] Bergmann *et al*. further developed this approach by proposing the Periodic Spatial-GAN (PSGAN), which extends the structure of the input noise distribution by constructing tensors with different configurations and dimensions.

SinGAN [107] introduced by Shaham *et al*. is also a single image model for texture generation, as PSGAN and our proposed models. SinGAN is designed to deal with natural images, and it is built as a pyramid of fully convolutional GANs, each in charge of learning the patch distribution at a different scale of the image. SinGAN can produce realistic image samples that consist of complex textures and non-repetitive global structures.

Achieving impressive results, InGAN was developed by Shocker *et al*. [112] and intended to capture the unique image-specific patch-distribution and map them to new images with different size and shapes that share with the source one the same patch distribution, which they loosely call "same DNA". They also proposed a new Encoder-Encoder architecture, called "Internal GAN", that encodes an image-specific internal statistics, which they claim provides a single unified framework for a variety of tasks.

These were works that influenced our proposed method. However, more recently, in One-Shot GAN [119], Tobias Hinz *et al*. proposed a two-branch discriminator, with content and layout branches designed to evaluate internal content regardless of scene realism. With this approach, One-Shot GAN avoids overfitting, which often happens when training a GAN in a low data scenario, like single image GANs. In ConSinGAN [50], the authors employed several stages of training, with different learning rates, aiming to control the balance between variance and conformity to the original training image. More straightforward, in [127] Yael Vinker *et al*., introduce a conditional GAN for learning to map from a primitive image representation (e.g. edges or segmentation) to an image. Its main contribution is introducing a new augmentation method that allows standard cGAN, trained on a single image, to train without overfitting. Finally, also an image-to-image translation method that can be trained on a single image, Taesung Park *et al*. present the Contrastive

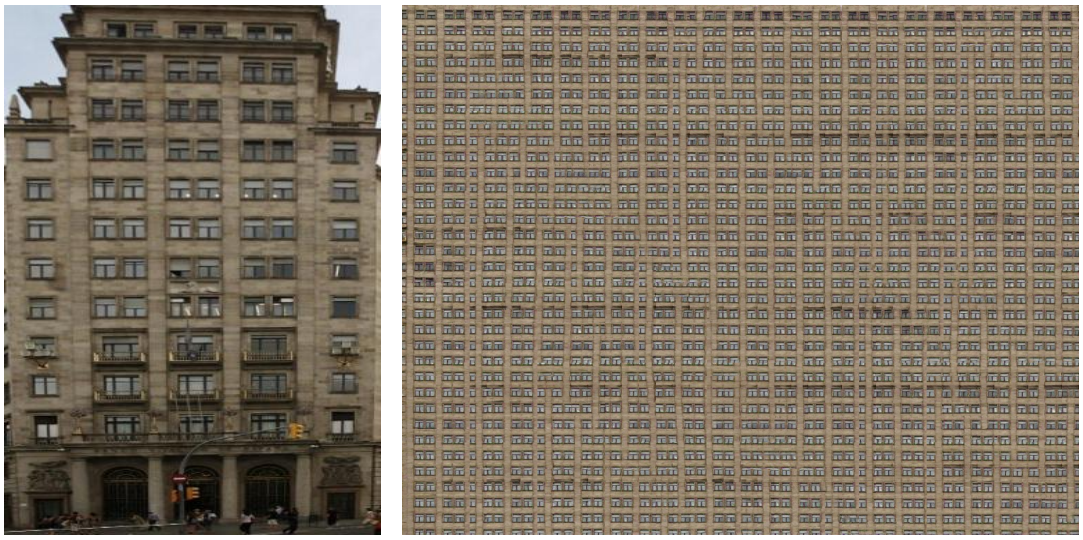
Unpaired Translation (CUT) in [91], a framework based on contrastive learning [89] that endeavours to encourage two elements (corresponding patches) to map to a similar point in a learned feature space, relative to other elements (other patches) in the dataset. This approach targets a way of keeping correspondence in content by maximising the mutual information between the corresponding input and output patches.

4.2 Methodology

As a continuation of the work presented in chapter 3, here we investigate a generative method that synthesises high-resolution images based on a single constraint source image. However, contrary to the QuiltGAN method presented in the previous chapter that depends on three GANs to synthesise samples, our current strategy consists of a single *conditioned Deep Convolutional Generative Adversarial Networks* (cDCGAN) trained to generate samples of an image patch conditioned on the surrounding image regions. In order to achieve this, we design a novel GAN architecture that is randomly conditioned by a subset of pixels in the input image and show how such an architecture can be used to progressively *grow* a realistic high-resolution image. Finally, we demonstrate how the synthesising process can be guided by carefully selecting the input seed, as outlined in Subsection 4.3.2.

4.2.1 Problem Formulation

Using just one image for training, we would like to get as a result a synthesised image that is different from the reference image used for training, but at the same time shares its main features. As an example, we used the facade [103] in Figure 4.1a to train our rcGAN and we would like, at the end of our two-steps pipeline, to get an image like the one in the Figure 4.1b. Figure 4.1b was synthesised by our two-steps process, as outlined in Subsection 4.2.3.



(A) Source image (272 × 512 pixels). (B) Synthesised image (1280 × 1280 pixels).

FIGURE 4.1: **Large-scale image generation.** Reference image utilised to train our rcGAN method and a 1.6Mpx synthetic output.

4.2.2 Proposed Method

To synthesise arbitrary sized, high-resolution images, we developed a method built on deep convolutional generative adversarial networks (DCGAN) [101] and conditional GAN (cGAN) [87]. To train our rcGAN model, we extract patches from the source image I and use these patches for training and for conditioning our generator G. To condition our generator G we developed a new algorithm that *Conditions by Randomly Selecting (CRS)* pieces of a given image patch. These are presented as a condition to the generator G, which then generates a plausible *completion* of these patches.

The Condition by Randomly Selecting (CRS) algorithm extracts a condition sub-patch from an image patch. The algorithm takes as input an image patch, denoted by S, of size $N \times N$ and a randomly generated number r within the range $[w \dots N - w]$, where $w = \lfloor N/6 \rfloor$. It also requires an integer number p, which is uniformly sampled from the set P containing different ways of selecting parts of S as a condition sub-patch.

The set P consists of the following possible ways of sampling from S:

1. **Full patch**. The entire image patch S is selected as the condition sub-patch without any extraction.

2. **Horizontal up**. All pixels in S with coordinates (x, y) where $y > r$ are selected as the condition sub-patch.

3. **Horizontal down**. All pixels in S with coordinates (x, y) where $y < r$ are selected as the condition sub-patch.

4. **Vertical left**. All pixels in S with coordinates (x, y) where $x < r$ are selected as the condition sub-patch.

5. **Vertical right**. All pixels in S with coordinates (x, y) where $x > r$ are selected as the condition sub-patch.

6. **Vertical and horizontal**. This method combines horizontal and vertical erasing, resulting in a corner condition. The selection includes pixels in S based on the conditions mentioned in both the horizontal and vertical directions.

The algorithm then outputs the condition sub-patch C_p extracted from the original image patch S, according to the selected method specified by the integer p.

Our model requires a generator G that can complete a given patch in three ways: to the right, below, and in the corner, as explained in Section 4.2.4. To achieve this, we sample from the source patch using these 6 methods. We believe this approach enhances the generator's G completion ability by offering multiple options for sampling from the same patch. In addition, this can improve the generalisation capability of the generator G and prevent overfitting, which is essential for generating a high-quality synthetic image. The pseudo-code for the *(CRS)* algorithm is presented in Algorithm 1.

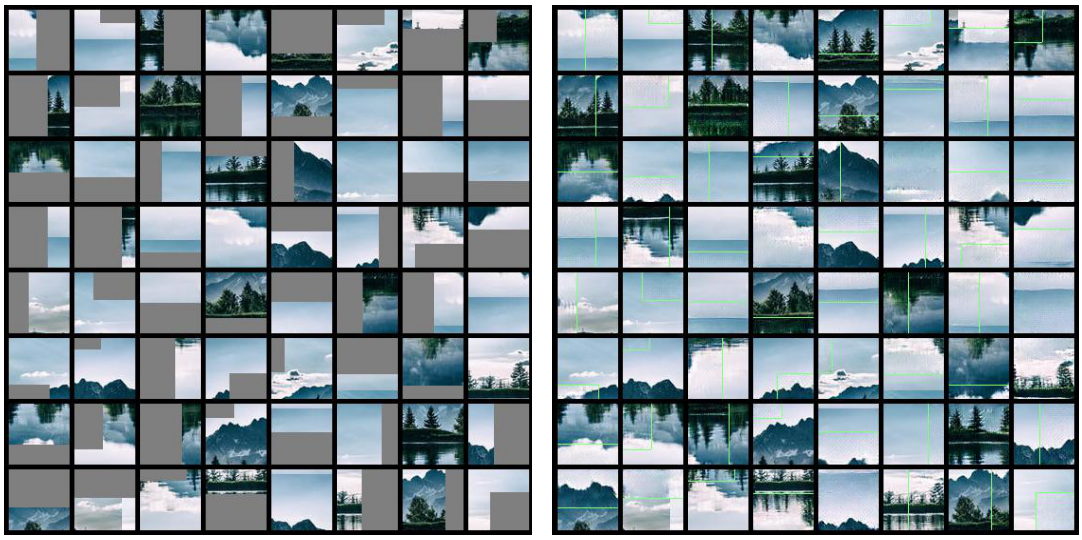
On every iteration of the training process a batch is processed by the *(CRS)* algorithm to generate condition sub-patches that will be presented to the rcGAN generator. Figure 4.2 is an example of our *(CRS)* algorithm in action.

4.2.3 Training a rcGAN

Our proposed rcGAN architecture consists of a non-conditioned discriminator D and a conditioned generator G. The generator takes as input a noise vector z sampled from a normal distribution concatenated with a $N \times N$ image patch S, previously processed by Algorithm 1, and then outputs another $N \times N$ image patch S^*.

Algorithm 1 : In the below pseudo-code, "pixels with x > r" means all pixels in S that have x-coordinate greater than r, and "pixels with (x < r) and (y > r)" means all pixels in S that have x-coordinate less than r and y-coordinate greater than r.

function CRS(*image patch* S)
  $p \leftarrow \text{random}(6)$
  $w \leftarrow \text{floor}(N/6)$
  $r \leftarrow \text{random}(w, N - w)$
  condition Cp
  if p == 1 **then**
    $Cp \leftarrow S$// Full patch
  end if
  if p == 2 **then**
    $Cp \leftarrow$ pixels with $y > r$ in S// Horizontal up
  end if
  if p == 3 **then**
    $Cp \leftarrow$ pixels with $y < r$ in S// Horizontal down
  end if
  if p == 4 **then**
    $Cp \leftarrow$ pixels with $x < r$ in S// Vertical left
  end if
  if p == 5 **then**
    $Cp \leftarrow$ pixels with $x > r$ in S// Vertical right
  end if
  if p == 6 **then**
    $Cp \leftarrow$ pixels with $(x < r)$ and $(y > r)$ in S// Vertical and horizontal
  end if
  return Cp
end function



(A) Before (B) After

FIGURE 4.2: **Conditioning by Randomly Selecting (CRS).** Examples of the six ways of generating conditions to our rcGAN, before and after merging the condition S with S^*. The green line on the right is there only to make it easier to visualise the inpainted area.

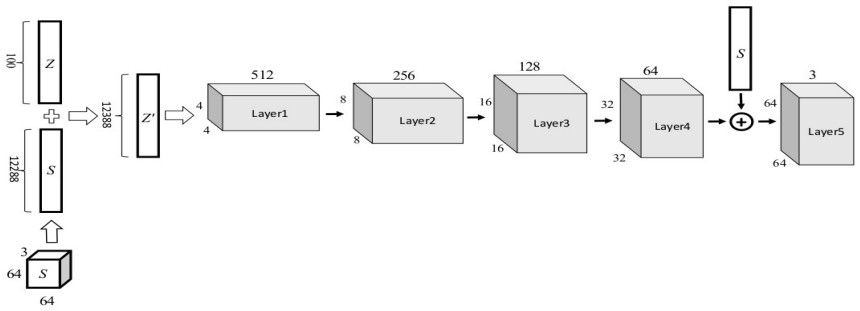


FIGURE 4.3: **Generator architecture.** Initially, the 64×64 input patch is reshaped and then concatenated with the input noise vector z. Finally, before being processed by the last layer, the output of the penultimate layer is concatenated with the condition vector S.

Instead of conditioning all layers, as in the previous chapter with the QuiltGAN method, we are just conditioning the first and last layers of the generator G. Empirical tests demonstrated that this approach speeds up the training process and doesn't compromise the quality of the generated patch S^*. For the first layer of the generator G, we concatenate the reshaped input image patch S with the provided noise vector before we feed it to this layer. For the last layer, we also concatenate the output of the penultimate layer with the reshaped input image patch S, and supplies this concatenation as input for this layer. See Figure 4.3 for the proposed generator network architecture.

In Figure 4.4, an input $N \times N$ image, processed by the Algorithm 1, is used to condition the generator. The $N \times N$ output of the generator S^* is merged with S

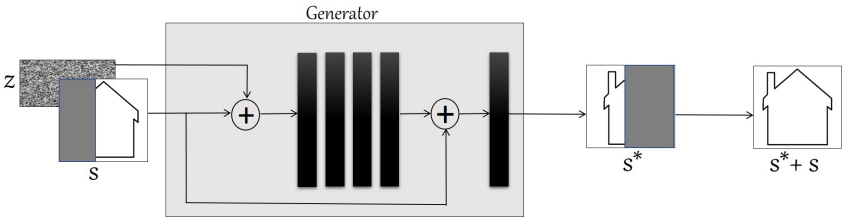


FIGURE 4.4: **Generator flow.** An input $N \times N$ image, processed by the Algorithm 1, is used to condition the generator. The $N \times N$ output of the generator S^* is merged with S before it's given as input for the discriminator.

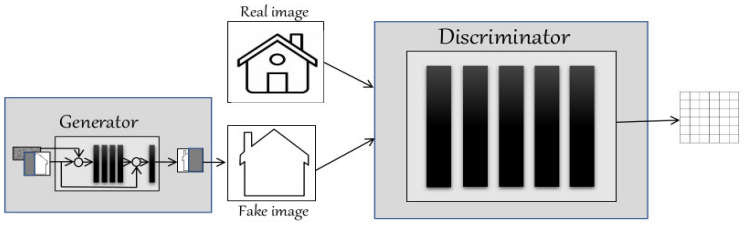


FIGURE 4.5: **Discriminator flow.** The discriminator receives as input a set of $N \times N$ patches and seeks to distinguish among those obtained from actual images and those resulted from merging S and S^*.

before it is given as input for the discriminator. The discriminator network D, see Figure 4.5, seeks to distinguish between the real and the generated image patches, while the generator network G strives to mislead it into misclassifying. Observe that our approach diverges from [87] as we are conditioning only the generator G, letting the discriminator D as proposed by [101], without any constraint. We took this decision because if the discriminator were also conditioned on the same information, it would have access to the same additional information as the generator, making it easier for the discriminator to identify generated samples as fake.

Following Shrivastava *et al.* in [114], instead of only outputting a probability of a patch S^* being real or fake, our discriminator D outputs a grid where each grid position corresponds to a local part of S^*, and represents the estimated probability that a local part of S^* is real or is being generated by G. This approach helps to improve the synthetic image quality and preserve the level of detail.

4.2.4 Patch Compositing

The second step of our rcGAN method is the generation of arbitrarily sized synthetic images. After training, our rcGAN recursively generates a sequence of samples which are then stitched together to synthesise a larger image. To stitch the GAN

generated patches, we propose a compositing algorithm, inspired by the *image quilting* algorithm proposed by Efros *et al*. [34], that works as follows:

To generate a synthetic image, with $M \times M$ blocks, each of them of size $N \times N$, we start by randomly selecting a patch S, or a *"seed"*, from the source image I, with size $N \times N/2$. This patch S is then passed to the generator G as a condition C_1. The right half of the generator output S^*, which produces a $N \times N$ output, is extracted and then stitched with S, forming the first block B_1 of size $N \times N$. To generate the left half part of block B_2, we take B_1 right half part as a condition C_2 and again give this patch to the generator G. The right half of the generator output S^* is extracted and then stitched with B_1. We repeat this process until the first row, with M blocks of size $N \times N$ each. To better understand how this process works, please refer to Figure 4.6.

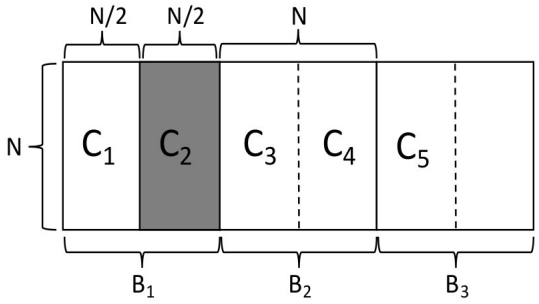


FIGURE 4.6: **Column conditioning.** The previously generated patch C_{i-1} is used as a condition to generate the current patch C_i, in grey, and so on.

To generate the subsequent rows the process is the same, but how the generator G is conditioned is different. For the first block $(i, 1)$ of a row $i > 1$, the half bottom part of the block $(i - 1, 1)$ is selected as a condition, Figure 4.7 shows an example.

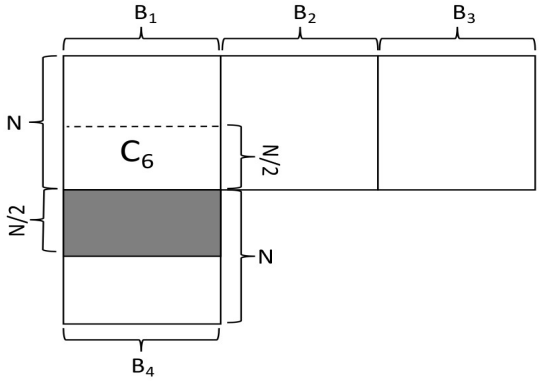


FIGURE 4.7: **Inner row conditioning.** How the condition of the first block of an inner row $i > 1$ is selected. The condition C_6 is used to generate the top half part, in grey, of block B_4 in this example.

For the subsequent blocks (i, j) in this row i, a corner condition is selected from the blocks $(i, j-1)$, $(i-1, j-1)$ and $(i-1, j)$. See Figure 4.8 for an example. With this strategy it is possible to synthesise images of any size.

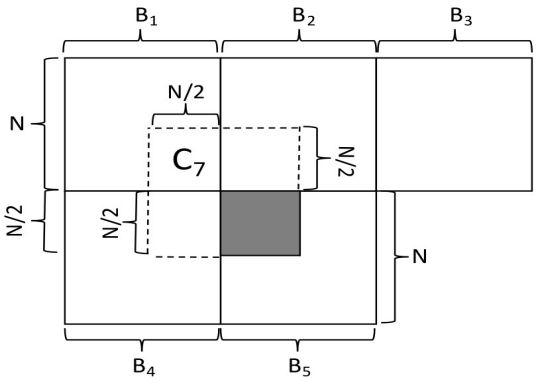


FIGURE 4.8: **Inner block in an inner row conditioning.** How the condition of the inner blocks in a inner row $i > 1$ is selected. To generate the top left part of block B_5, in grey, a corner condition is selected from the blocks B_1, B_2 and B_4.

4.3 Experimental Results

This section describes the experiments performed with our randomly conditioned GAN (rcGAN) method as proposed in 4.2.2. To evaluate the effectiveness of our method, here we compare our rcGAN framework with three selected methods that, like ours, are also generative models that can learn from a single image:

1. *SinGAN* [107], which is built as a pyramid of fully convolutional GANs, each in charge of learning the patch distribution at a different image scale.

2. *InGAN* [112], which intends to capture the unique image-specific patch distribution and map them to new images with different sizes and shapes that share the same patch distribution as the source one;

3. *Spatial-GAN* [58], which proposed extending the input noise distribution space from a single vector to a spatial tensor.

With these experiments, we intend to evaluate our algorithm against other methods and demonstrate our rcGAN framework's strength.

4.3.1 Experimental Setup

For all experiments executed with our rcGAN model, the block size N was 64 pixels, the size of the rcGAN input and synthesised patches. As in Chapter 3, all experiments were conducted on a single GPU NVIDIA GeForce GTX 1070 with 8 gigabytes of RAM, but now, one of our method's advantages is that we do not need to train three GANs to get the results. Instead, we use a single GAN that takes around 12 hours to be trained on a single image, which is less than half of the time taken in the previous Chapter if we consider that in Chapter 3, we have to train the GANs sequentially due to lack of resources.

In order to provide a comprehensive evaluation of our method, we compared it against three alternative approaches: SinGAN, InGAN, and Spatial-GAN. To ensure a fair comparison, we utilised the default parameters as suggested by the respective authors for each of these methods. This approach ensures consistency and allows for a direct comparison of the performance and effectiveness of our proposed method against these established techniques.

Furthermore, we strengthened the demonstration of our approach by integrating supplementary facade images, leveraging the superior efficiency of our rc-GAN over the Quilt-GAN method discussed in Chapter 3. This improved efficiency arises from not having to train three distinct GANs for generating synthetic images. Moreover, our method reduces the potential for errors by eliminating the need to synchronise the outcomes of three separate GANs.

The results of our proposed two-steps method for a wide variety of input images are presented in Figure 4.10. Column two was created using SinGAN; column three is InGAN generated, column four was generated with Spatial-GAN and column five was created using our two-steps method as outlined in 4.2.3.

4.3.2 Evaluation

We notice that SinGAN can only properly handle natural images like the one in Figure 4.10 row two. For the other types of images, SinGAN generated samples that lost the overall structure presented in the source image, *e.g.* the apple in row one. The facades also lost the alignment of the windows and balcony railings. Spatial-GAN

unsuccessfully handles most images presented to it and cannot preserve the existing spatial information in the facades; except for the input in row ten, where it was capable of generating aligned windows. Regarding texture generation, Spatial-GAN was effective in creating high-quality textures like the ones in rows two and four. InGAN delivered impressive results and can handle all types of pictures shown, and as our method can synthesise images of any size but suffer from lack of flexibility. Examples can be seen in row seven, where we could get rid of the tree in the middle and also the light pole at the top. At row ten, we could also generate an image that does not have the tree and the light pole presented in the input image on its right. In row eleven, our model could produce a new shadow pattern. Finally, in rows twelve and thirteen, we could avoid the grey areas by manually selecting the seed during the generation process. To visualise how this can be done, see Figure 4.9 for an example of how seed selection can be used to avoid undesired features in the synthesised image.

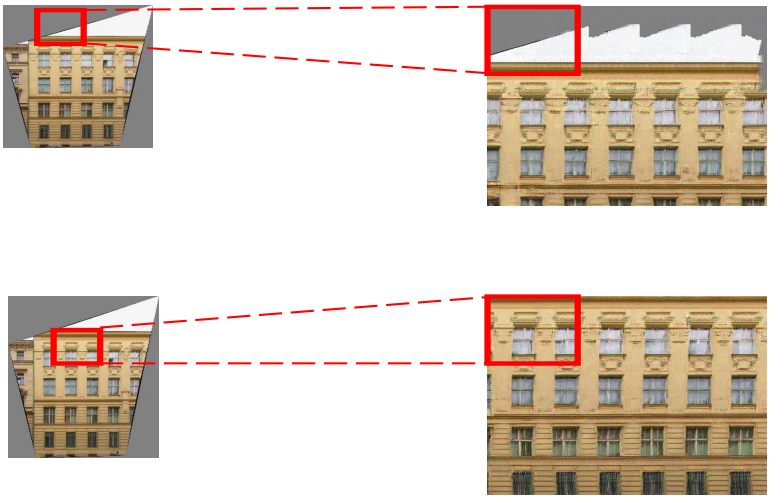


FIGURE 4.9: **rcGAN seed selection.** Above, the seed was selected randomly, and the synthetic image has the non desired grey area on top. Bellow, the seed was chosen manually intending to avoid the grey area to be synthesised in the output image.

One case of failure of our method results in a sort of *degenerated* condition. If our rcGAN generated patch S^* has some errors, *i.e.* some non-realistic appearance, then the next patch generated which is conditioned on it, is extra deteriorated. This results in a succession of progressively worse image patches driving to a resulting unrealistic image. See Figure 4.11 for examples. This is resultant from the recursive application of the rcGAN patches that follows a linear path through the image.

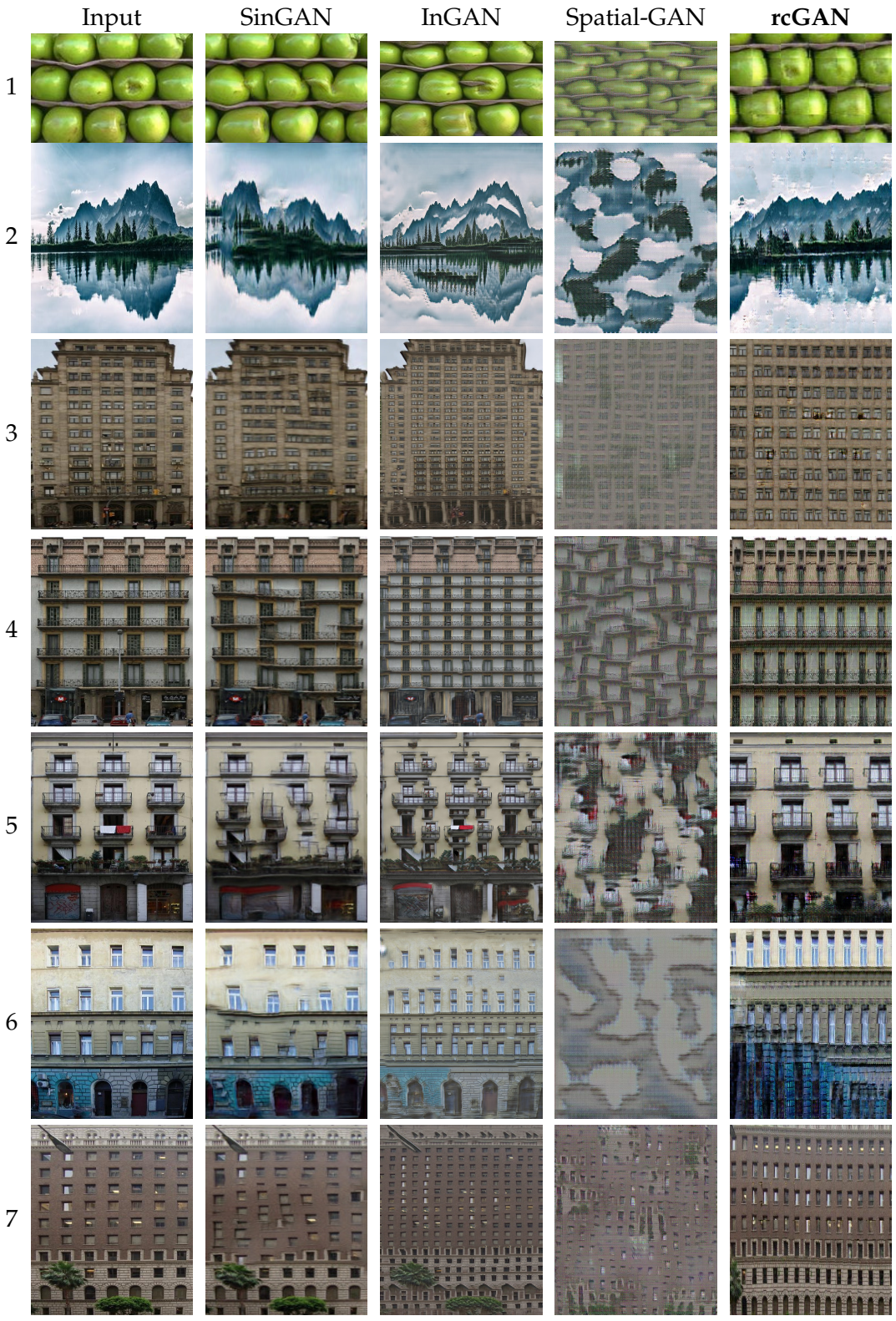


FIGURE 4.10: **Image Synthesis:** The images size varies according to model output.

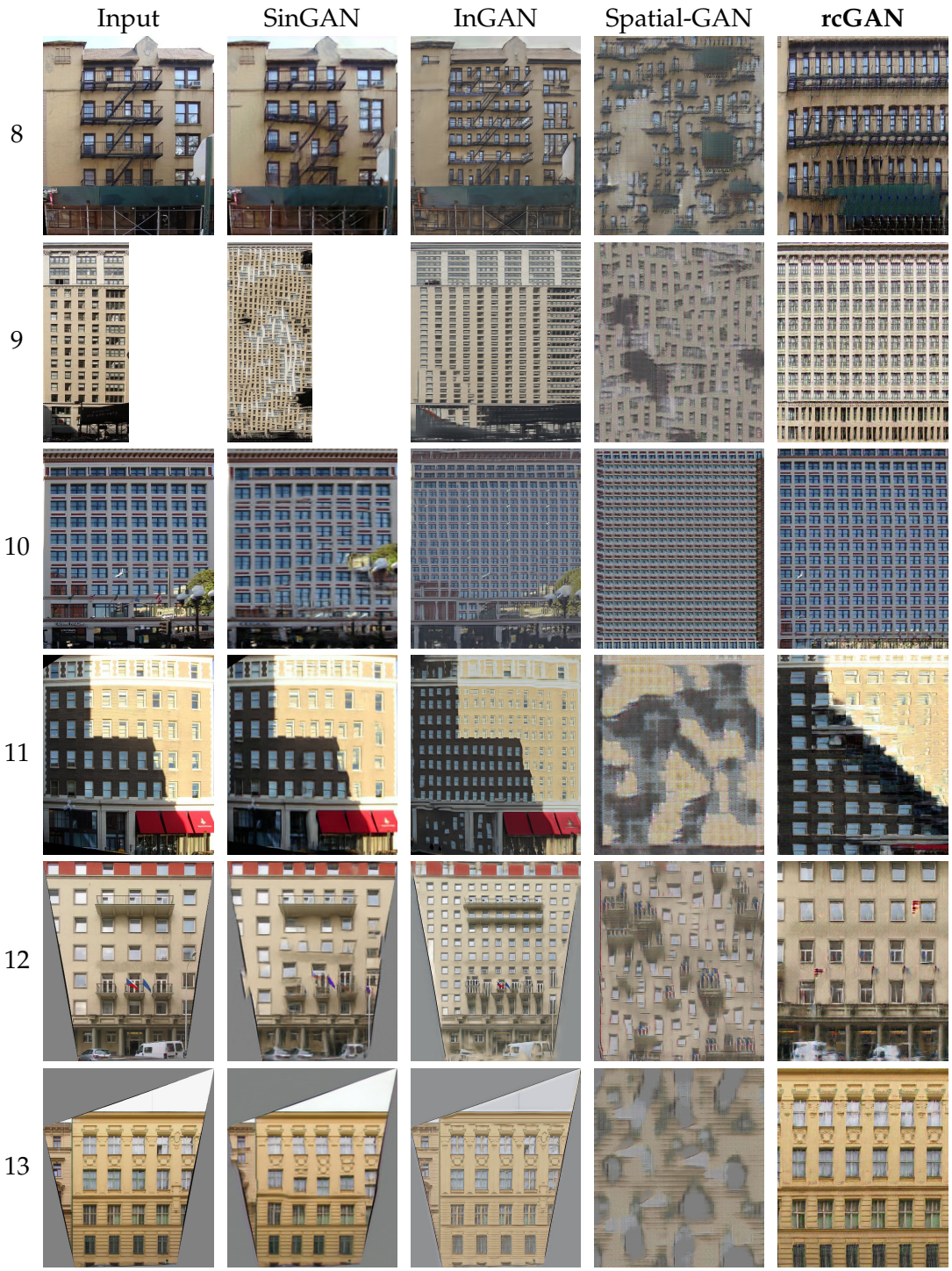


FIGURE 4.10: **Image Synthesis (cont.) :** The images size varies according to model output.

We hypothesise that to prevent image degradation caused by the recursive application of the rcGAN, a few methods can be considered:

1. *Selective patch generation*: Implementing a mechanism to selectively generate patches based on their quality can prevent the propagation of unrealistic appearances. By evaluating the realism of each patch generated by the rcGAN and discarding those with significant errors, the subsequent patches will not be conditioned on degraded inputs, reducing the overall image degradation.

2. *Feedback mechanism*: Introducing a feedback mechanism that allows the rcGAN to learn from its own mistakes can be beneficial. By providing the generator with information about the quality and realism of its output patches, it can adjust its generation process accordingly. This feedback mechanism can incorporate a separate discriminator network that evaluates the generated patches and provides feedback to the generator, encouraging it to produce more realistic and accurate patches.

By implementing these or similar strategies, we believe it is possible to mitigate the image degeneration issue caused by our method's recursive application, resulting in improved image generation quality and preventing the successive deterioration of patches.

FIGURE 4.11: **A mode collapse case.** To create the image, the rcGAN is used in a recursive and linear manner, which can sometimes lead to errors accumulating. These particular examples are based on images from rows 9, 7, and 2.

4.4 Quantitative Evaluation

We used a metric called the Single-Image Version of the Fréchet Inception Distance to confirm how realistic our synthetic images were and if they accurately represented

the training image's internal statistics. This metric was developed by the authors of SinGAN [107].

Fréchet Inception Distance (FID) [49] serves as a commonly used metric for evaluating GANs. It quantifies the disparity between the distribution of deep features in generated images and that of real images. As in our scenario, SinGAN operates with only one training image. Hence their proposal of a single image metric based on FID. Named Single Image FID (SIFID), this metric focuses on the internal training image patch statistics, rather than extracting information from a whole dataset. Instead of utilising the activation vector following the final pooling layer in the Inception Network [121], SIFID employs the internal distribution of deep features at the output of the convolutional layer just before the second pooling layer. Thus, SIFID calculates the FID between the statistics of these features in the real image and the generated sample.

The results of our rcGAN approach are displayed in Table 4.1. Table 4.1 shows that the InGAN method had the highest SIFID score, with eight out of thirteen being the best scores. It's worth noting that lower scores are better. We hypothesise that InGAN's ability to generate images that accurately maintain semantic information, such as window alignment, contributed to its success.

As anticipated, Spatial-GAN produces images that are outside the range of the training image distribution, resulting in the worse scores in all instances.

Although our approach may not produce always optimal quantitative results, we believe that our method remains stronger compared to the other methods discussed in this context. This is attributed to our ability to deliver increased variability while preserving the original distribution, along with providing a level of control over the generated synthetic image by allowing manual selection of the seed.

4.5 Summary

In this chapter, we proposed a two-steps process to generate images, which is built on a randomly conditioned DCGAN (rcGAN). Our process uses only a single reference image to train our rcGAN to create image patches that will be later on stitched together to synthesise an arbitrary sized image. The size of the resultant image is

Input	SinGAN	InGAN	Spatial-GAN	rc-GAN
1	0.070	0.596	0.798	**0.061**
2	0.668	**0.363**	0.526	0.382
3	0.305	**0.115**	0.914	0.242
4	0.429	**0.184**	0.800	0.376
5	0.260	**0.174**	0.911	0.269
6	**0.140**	0.161	0.943	0.254
7	0.240	0.336	0.908	**0.073**
8	0.464	**0.383**	0.510	0.615
9	0.301	**0.167**	0.911	0.226
10	0.572	0.310	0.745	**0.143**
11	0.315	0.224	0.917	**0.095**
12	0.669	**0.299**	0.481	0.481
13	0.551	**0.109**	0.730	0.388

TABLE 4.1: **Experimental results**. The Single Image FID (SIFID) score evaluates each method's performance. A lower score indicates better performance. Each Input number corresponds to an image with the same Input number in Table Figure 4.10.

constrained by the number of generated image patches that are stitched together to create it. Our rcGAN is conditioned on randomly selected adjacent parts of the image and learn how to complete it by producing a *matching* image area. We compare the results of our two-steps methods with *SinGAN* [107], *InGAN* [112] and *Spatial-GAN* [58]. We conclude that despite the simplicity of our framework, as we use just a single DCGAN, our proposed technique can handle any input image: textures, natural landscapes and facades. Our framework also compares very favourably against the state of art techniques. Additionally, it provides more control over the synthesised image features, as we can influence the process by manually selecting the seed, which is a capability that the other methods we compared against do not offer.

Part III

Deep Learning For Data Augmentation

Chapter 5

Cycle And Semantic Consistency For Data Augmentation

The sparsity of training data can hinder the performance of supervised machine learning algorithms, which often require large amounts of data to train and avoid overfitting. Typical deep neural networks have hundreds of millions of parameters to learn, which requires many passes over the training data. Running a large number of iterations on small datasets can result in overfitting, which is usually remedied by one or more of the following: acquiring more data, applying regularisation and performing data augmentation. The latter approach is limited chiefly to simple randomised manipulation of an existing dataset (*e.g.* affine warping, rotation or other perturbations) [113].

Founded on our previous experience of using GANs for image synthesis, in this chapter, we extend the work presented in Part II by addressing the research questions 2 and 3 (outline in section 1.2) and investigate to what extent an image-to-image translation model like CycleGAN [140] can be used as a data augmentation method.

The image-to-image translation is a computer vision problem that involves learning how to map an image from one domain A to another domain B using a training dataset. However, this translation is not always accurate, and relevant semantic information can deteriorate during the translation process. To handle this problem, we propose a new Cycle and Semantic Consistency (CSC-GAN) model, an adversarially trained image-to-image translation method that strengthens local and global

structural consistency through cycle consistency and a new loss function that is constrained by semantic segmentation. Our formulation encourages the model to preserve semantic information during the translation process. For this purpose, our loss function evaluates the synthetically generated image's accuracy against a previously trained semantic segmentation model.

We build on the success of image-to-image translation to propose an approach to data augmentation that applies to semantic segmentation tasks. Under our scheme, an arbitrary number of new images are generated by 'translating' each ground truth label image in our training dataset. By augmenting that dataset with the resulting image/labelling pairs, it is common to observe an enhancement in the accuracy of the semantic segmentation model.

The results reported in this chapter show that our proposed method can significantly increase the level of details in synthetic images. Furthermore, we apply our CSC-GAN to a highly sparse building facade dataset [103] that consists of only 606 images and labels. Our experiments verify that the image-to-image translation approach strengthened by our new semantic loss can be successfully used as an augmentation method to improve pixel-level semantic segmentation models' accuracy.

The remainder of this chapter is organised as follows: The current literature review on adversarial methods for image synthesis is presented in Section 5.1, the proposed methodology is detailed in Section 5.2, and the experimental results are presented in Section 5.3. Finally, conclusions are summarised in Section 5.4.

5.1 Related work

The GAN framework was first introduced to generate visually realistic images and, since then, many applications have been proposed, including data augmentation [113], a technique widely used to increase the volume of data available for training.

One of the areas where GANs have been employed for data augmentation is Medical Imaging. Yi *et al*. in [134], surveyed 150 published articles in the medical image synthesis area and found that GANs are employed for image reconstruction, segmentation, detection, classification and cross-modality synthesis. The main reason for this widespread use seems to be the relative sparsity of labelled datasets in

the highly specialised medical image domains.

The same conclusion is reached by Bowles *et al*. [15], who investigate the use of GANs for augmenting CT scan data. They use a Progressive Growing of GANs (PGGAN) network [63] to generate the synthetic data in the joint image-label space. Their results show that GAN augmentation works better, when the dataset is more sparse. Unfortunately the PGGAN framework is more suitable for spatially registered datasets (e.g. faces, or medical imaging). Using an extensive image dataset, Sandfort *et al*. [108] trained a CycleGAN model [140] to transform contrast Computed Tomography (CT) images into non-contrast images, for data augmentation in CT segmentation tasks. According to the authors, the publicly available datasets consist universally of contrast-enhanced CT images, while real-world data contains a certain percentage of non-contrast CT images. This domain shift affects the performance of real-world applications negatively. Using CycleGAN to alleviate this issue, the authors report significant improvement in CT segmentation tasks performance.

When the available data is not uniformly distributed between the distinct classes, the accuracy of an image classification model can degenerate. Mariani *et al*. propose a balancing generative adversarial network (BAGAN), an augmentation method that can generate new minority-class images and then restore the dataset balance in [85], . Likewise, Antoniou *et al*. proposed a Data Augmentation Generative Adversarial Network (DAGAN) architecture based on a conditional GAN, conditioned on images from a given class c, where the generator network also uses an encoder to project the condition down to a lower-dimensional manifold [3]. Their adversarial training leads their system to generate new images from a given sample, one that appears to be within the same class but look different enough to be a diverse sample.

An impressive dataset augmentation method using GAN for semantic segmentation is introduced by Richter *et al*. [2]. They present an approach for creating a semantic label for images extracted from modern computer games. Using game engines to generate endless quantities of labelled data is a longstanding theme in Computer Vision research. Their experiments show that using the acquired data to supplement real-world images significantly increases accuracy, therefore a network trained on unrealistic data can generalise very well to existing datasets.

CyCADA, proposed by Hoffman *et al*. in [51], also explores ways of enforcing

(a) Real label (b) Real Image (c) CycleGAN (d) CSC-GAN (ours)

FIGURE 5.1: **Translation of facade label to a facade image.** The task is to translate from label to image. (a) and (b) are the real label and the real image, (c) is CycleGAN generated, that incorrectly translates the facade behind the tree and the balcony railings. (d) Using our CSC-GAN, that implements semantic consistency in the translation process, the result is a more realistic image.

semantic consistency on image synthesis. This adversarial unsupervised adaptation algorithm aims to learn a model that correctly predicts a label for a target data. Their input is the source data X_S, the source label Y_S and the target data X_T. Their aim is to learn a model that can correctly predict the label Y_T for the target data X_T. In contrast, our work presented in this chapter doesn't seek to predict but to increase the translated image quality with the help of a pre-defined classifier. We are provided with the source data X_S and the source label Y_S. Our purpose is to learn a model that does the translation $Y_S \rightarrow X_S$ more accurately.

5.2 Methodology

In this section, we present our *Cycle and Semantic Consistent GAN* (CSC-GAN) framework. We consider the problem of image-to-image translation with cycle and semantic consistency, where we are provided with the source data X and source labels Y. The aim is to learn a stochastic model f that translates a labelling into the corresponding image (*i.e.* $Y \rightarrow X$) in such a way that the resultant images are so realistic they can improve the results of a deep semantic segmentation model [20, 81], when used as a dataset augmentation technique. To do so, we extend the CycleGAN framework by adding a new loss function $\mathcal{L}_{sem}$ that evaluates how accurate the synthetic generated image is against a previously trained semantic segmentation model g. See figure 5.1 for an example of our method.

To establish the background, we first review in the following sections, the CycleGAN models on which our method is based.

5.2.1 CycleGAN Loss Functions

The CycleGAN full objective is composed of an adversarial loss and a cycle consistency loss, as follows.

Adversarial Loss

The adversarial loss is given by

$$\begin{aligned}\mathcal{L}_{GAN}(G, D_Y) = \mathbb{E}_{y \sim p_{data}(y)}[\log D_Y(y)] \\ + \mathbb{E}_{x \sim p_{data}(x)}[\log(1 - D_Y(G(x)))].\end{aligned} \tag{5.1}$$

where G tries to generate images that look similar to images from domain Y and D_Y aims to distinguish between the translated samples $G(x)$ and real samples y. G aims to minimise this objective against an adversary D that tries to maximise it.

Cycle Consistency Loss

Adversarial losses individually cannot guarantee that the learned function can map a single input x to a desired output y. To additionally decrease the space of possible mapping functions, CycleGAN authors argue that the learned mapping functions should be cycle-consistent, intending to encourage the source content to be preserved during the conversion process. For each image x from domain X, there exists a different map F that is able to bring x back to the original image, *i.e.*, $x \rightarrow G(x) \rightarrow F(G(x)) \approx x$. This is called the *forward cycle consistency*. Similarly, for each image y from domain Y , G and F should also satisfy *backward cycle consistency*: $y \rightarrow F(y) \rightarrow G(F(y)) \approx y$. This behaviour is encouraged using a cycle consistency loss:

$$\begin{aligned}\mathcal{L}_{cyc}(G, F) = \mathbb{E}_{x \sim p_{data}(x)} \parallel F(G(x)) - x \parallel_1 \\ + \mathbb{E}_{y \sim p_{data}(y)} \parallel G(F(y)) - y \parallel_1 .\end{aligned} \tag{5.2}$$

Full Objective

The CycleGAN full objective is

$$\begin{aligned}\mathcal{L}(G, F, D_X, D_Y) = &\mathcal{L}_{GAN}(G, D_Y) \\ &+ \mathcal{L}_{GAN}(F, D_X) \\ &+ \lambda \mathcal{L}_{cyc}(G, F).\end{aligned} \tag{5.3}$$

where λ controls the relative importance of the two objectives.

5.2.2 Semantic Consistency Objective

As we have access to the source labelled data, we aim to encourage high semantic consistency after image translation explicitly. For this purpose, we pre-train a semantic segmentation model g, on the same *training* set used to train our CSC-GAN model, and use this model g to evaluate the synthetic images during the CSC-GAN training. By fixing the model g weights during the CSC-GAN training, we guarantee that a good segmentation result, obtained from a synthetic image, is due to an improvement in the synthetic image quality, as the model g was trained on real images. Using the segmenter model g we propose our semantic consistency objective as

$$\mathcal{L}_{sem}(g, F) = \mathbb{E}_{y \sim p_{data}(y)} \mathcal{L}_{cs}[g(F(y)), y]. \tag{5.4}$$

where $\mathcal{L}_{cs}[.,.]$ is the cross-entropy loss [51] comparing two segmentation masks $g(F(y))$ and y. This equation means that given a label $y \in Y$, the model g predicts the labels for $F(y)$, *i.e.*, the synthetic image generated using the real label y. The loss $\mathcal{L}_{sem}$ evaluates this prediction, and its result is added to the CycleGAN objective function, intending to help improve the overall synthetic image quality. It is interesting to note the superficial similarity between the semantic consistency loss $\mathcal{L}_{sem}$ and the second term of the cycle consistency loss (Eq. 5.2). Both G and g map from images to labels but (a) g is producing 1-hot encoding per label while G produces an RGB image as in [140], hence the different choice of image distance metric and (b) g is a pre-trained network (we use the DeepLabV3 [20] architecture) while G is a network

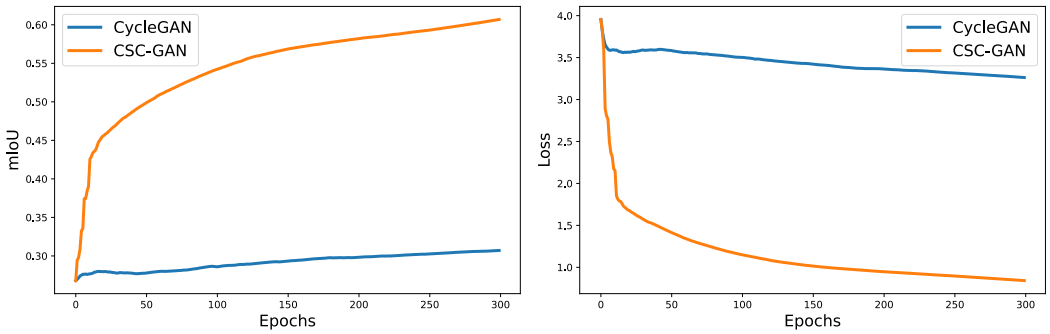


FIGURE 5.2: **Semantic Loss Effect.** Impact of the proposed semantic consistency loss over the synthetic images during model training.

we train adversarially. We experimented with removing the trainable G network but were unable to achieve a converged solution, possibly because G introduces a convex relaxation to the optimisation problem. The full objective is then

$$\begin{aligned}\mathcal{L}(G, F, D_X, D_Y) = \mathcal{L}_{GAN}(G, D_Y) \\ + \mathcal{L}_{GAN}(F, D_X) \\ + \lambda \mathcal{L}_{cyc}(G, F) \\ + \mu \mathcal{L}_{sem}(g, F).\end{aligned} \tag{5.5}$$

where λ and μ are relative important weights. To show the impact of the new semantic consistency loss, Figure 5.2 presents the evaluation comparison of the synthetic images, during model training, with and without our proposed loss. It shows a dramatic improvement in the mIoU score and the cross-entropy loss when the objective described in equation 5.5 is used, instead of the regular CycleGAN loss defined in equation 5.3.

5.3 Experimental Results

This section presents the attained results of our CSC-GAN approach compared against the Facade dataset [103] augmented with the regular CycleGAN model [140], SPADE model [92], which is a state-of-art label to image translation and two style transfer models [40, 57], the latter being designed explicitly for dataset augmentation. The datasets used in the experiments are listed in Table 6.1.

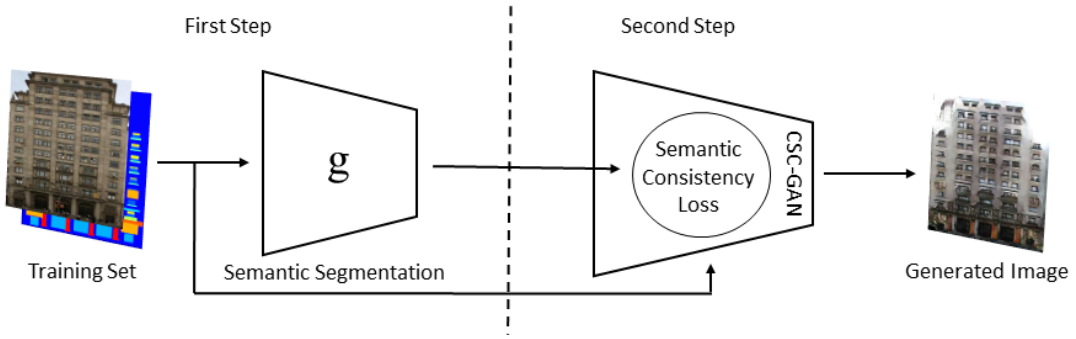


FIGURE 5.3: **CSC-GAN two steps.** First we train a semantic segmentation model g on the *training* set, then we use it to evaluate the synthetic images quality during our model training.

5.3.1 Experimental Setup

In order to compare these augmented datasets, for each one we trained two semantic segmentation models, DeeplabV3 [20] and Fully-Convolutional Network (FCN) [81], both with a ResNet101 [47] backbone. Each model is trained for 50 epochs with a learning rate set to 0.0002 and the Adam optimiser [64]. The performance of each model is reported using the mean intersection-over-union (mIoU) score [111] evaluated over the *test* set. The Mean Intersection over Union (mIoU) is a commonly used evaluation metric in the field of computer vision, specifically for tasks related to semantic segmentation. It is used to assess the accuracy and quality of predicted segmentation masks or regions compared to ground truth annotations.

The facade dataset [103] has been randomly split into a *training* X_S, with 80% or 484 images and a *test* X_T set, with 20% or 122 images. As shown in Table 5.1, there are six variations of the *training* set, but the *test* set remains the same across all experiments.

The augmentation is done as follows: For each label $y \in X_S$ a synthetic image $x = F(y)$ is generated and the pair (y, x) is added to the dataset. By the end a new *training* set $X_S^{'}$ is created of twice the size of X_S. This is done because one synthetic image per each label $y \in X_S$ is added to the new *training* set $X_S^{'}$.

The CSC-GAN model is trained in two steps: First, we train the semantic segmentation model g on the regular facade dataset, which is used as a labeller during the CSC-GAN training, then we train the CSC-GAN model, as described above. Figure 5.3 illustrates these two stages.

Dataset Name	Size	Description
facade	484	regular training set without any augmentation. [103]
facade+CycleGAN	968	regular plus images from regular CycleGAN model [140].
facade+SPADE	968	regular plus images from SPADE model [92].
facade+Styleaug	968	regular plus images from style augmentation model [57].
facade+Arbitrary	968	regular plus images from arbitrary artistic stylization model [40].
facade+CSC-GAN	968	regular plus images from our CSC-GAN model.

TABLE 5.1: **Datasets.** Lists of all datasets used in the semantic consistency experiment.

		DeeplabV3 [19]		FCN [81]	
Dataset Name	**FID**	**mIoU**	**Accu.**	**mIoU**	**Accu.**
facades [103]		0.552 (0.016)	0.706 (0.013)	0.558 (0.003)	0.711 (0.002)
facade+CycleGAN [140]	**22.742**	0.557 (0.008)	0.710 (0.007)	0.562 (0.002)	0.714 (0.002)
facade+SPADE [92]	24.107	0.565 (0.009)	0.716 (0.008)	0.572 (0.004)	0.723 (0.003)
facade+Styleaug [57]	75.307	0.543 (0.005)	0.698 (0.005)	0.561 (0.005)	0.714 (0.004)
facade+Arbitrary [40]	37.804	0.550 (0.005)	0.703 (0.004)	0.562 (0.003)	0.714 (0.002)
facade+CSC-GAN (ours)	29.268	**0.573 (0.005)**	**0.723 (0.004)**	**0.574 (0.003)**	**0.724 (0.002)**

TABLE 5.2: **Experimental results**. Accu. is the pixel accuracy; the FID [49] metric is calculated against the dataset without augmentation. Each experiment was executed 5 times, the mIoU and the pixel accuracy results reported are the mean and the standard deviation for these 5 executions.

Table 5.2 presents the results. Each experiment was executed 5 times and the results reported are the mean and the standard deviation for these 5 executions. We decided to execute 5 times because we verified that the standard deviation among the experiments was very low. When the standard deviation is very low, it implies that the variability in the results is minimal, and the measurements are highly consistent. In such cases, executing the experiment 5 times can be considered sufficient, given that the low standard deviation indicates a high level of precision.

In our experimental setup, we trained both semantic segmentation models, i.e. DeepLabV3 and FCN, and the generative models, i.e. CycleGAN, SPADE and our CSC-GAN, on a single GPU NVIDIA GeForce GTX 1070 with 8 gigabytes of RAM. Training each semantic segmentation model requires approximately 5 hours, and the generative models demanded roughly 8 hours each. Consequently, training all three generative models consumed close to 24 hours in total. Training both semantic segmentation models takes a total of 10 hours. Hence, the training process for the semantic segmentation models alone took roughly 50 hours to complete. By adding the 24 hours required for the generative models, the entire experiment encompassed approximately 74 hours to finish, all while utilising a single GPU with only 8 gigabytes of RAM.

5.3.2 Evaluation

The presented results show that the dataset augmented with images from our CSC-GAN model can outperform the regular facade dataset by approximately 4% when the DeeplabV3 [19] model is trained on it and by roughly 3% with the FCN [81] model, as shown in the last row on Table 5.2. SPADE also provided promising results, but it was also outperformed by our method. The datasets augmented by [57] and [92] got a decrease in performance concerning the regular Facade dataset, showing that style transfer is not a good technique for dataset augmentation on a per-pixel classification scenario as semantic segmentation.

Analysing the FID [49] score in the first column suggests that a lower FID value does not necessarily imply that the synthetic image is superior in terms of data augmentation, and it is not necessarily correlated with the mIoU, as shown in Figure 5.4.

The FID is a metric commonly used to assess the quality of generated or synthetic images. It measures the similarity between the statistical properties of the generated images and real images by comparing feature representations extracted from a pre-trained Inception model [49]. Generally, a lower FID value indicates a higher similarity between the generated and real images, implying better image quality.

On the other hand, mIoU is a metric used to evaluate the performance of image segmentation algorithms. It quantifies the overlap between the predicted and ground truth segmentation masks for various object classes. A higher mIoU value indicates better accuracy in capturing the object boundaries and overall segmentation quality.

The graph in Figure 5.4 suggests that despite a lower FID value, which indicates a higher similarity to real images, it does not necessarily translate into better performance in terms of mIoU. In other words, the visual quality of synthetic images, as measured by FID, does not guarantee their effectiveness in improving segmentation performance, as measured by mIoU. This indicates a lack of correlation between the FID and mIoU metrics, in this scenario.

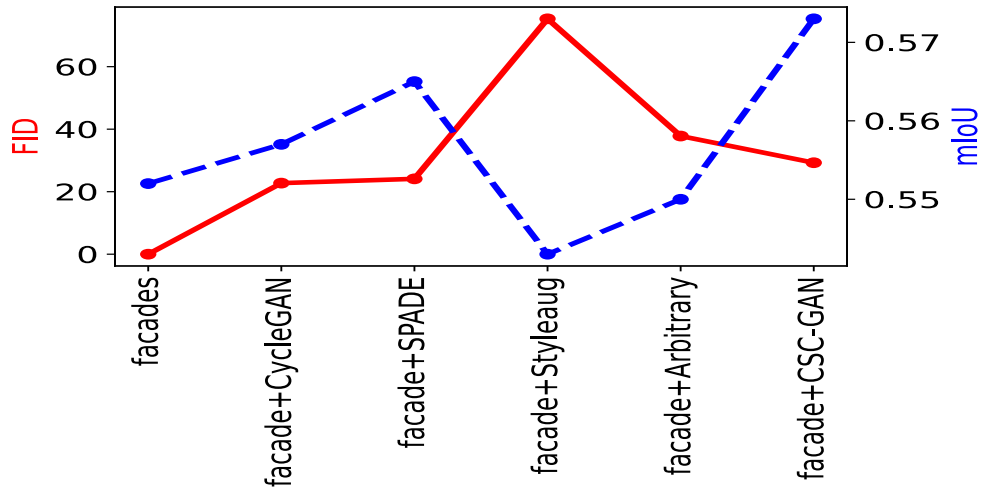


FIGURE 5.4: **FID x MioU.** The *Frechet Inception Distance* (FID) is a metric that compares statistics of real and generated images and summarises how similar the two groups of images are. A lower FID score indicates that the two groups of images are more similar. Based on the graph, it appears that having a lower FID score does not necessarily indicate that the synthetic image is more effective for data augmentation.

This information implies that while FID can provide insights into the visual fidelity of synthetic images, it may not be the exclusive factor to consider when evaluating their suitability for data augmentation in tasks like image segmentation.

Why does this happen? This finding is an idea we will delve further into in the next chapter, and the answer is related to our research question 4.

To better understand the improvement brought by our CSC-GAN, in Figure 5.5 we compare the correctly predicted pixels of the models trained with CycleGAN and CSC-GAN augmented datasets with the model trained with the regular facade dataset. The comparisons are made over the test set segmentation results.

Figure 5.5 presents the comparison of the regular facade with CycleGAN; the graph shows that for six out of twelve labels, there is a drop in the correctly predicted pixels, *cornice* and *decoration* showing the most substantial difference. Conversely, except for *background*, *window* and *shop* the model trained with our CSC-GAN augmented dataset attained a substantial improvement in the correctly predicted pixels. In particular, the *decoration* and *balcony* labels, are two very challenging categories where our augmented datasets substantially improved over the facade dataset. The reason is the increase in level-of-detail obtained by the CSC-GAN model to the generated images, as shown in Figure 5.6.

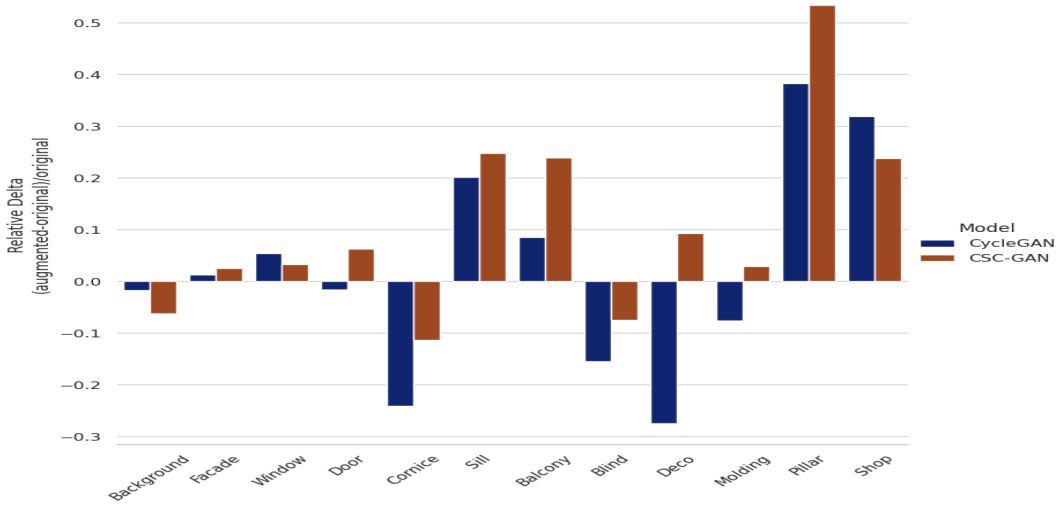


FIGURE 5.5: **Relative delta.** This chart shows the number of accurately predicted segmented pixels for both the standard facade dataset and the augmented dataset. It shows that for the majority of classes, our CSC-GAN model can produce more accurate images for data augmentation.

In Figure 5.6, column (c), note that our model is capable of learning the appearance of balcony railings and in the second row the extended model begins to show the pillar as a 3D structure, instead of just a flat shape as shown in column (b). The third row, column (c), shows that the extended model can also learn the decoration, which the regular CycleGAN model represents as a flat shape.

5.4 Summary

Founded on our previous experience of using GANs for image synthesis, in Chapter 5, we extended the work presented in Part II by addressing research questions 2 and 3 (outline in section 1.1) and presented a cycle-consistent image generation framework that combines semantic constraints to deliver an increased level of detail in the generated images. Our formulation encourages the model to preserve semantic information during the translation process. This refinement enables using the proposed method for data augmentation in the semantic segmentation domain. By introducing a new loss function that is constrained by semantic segmentation, we were able to improve the performance of the Cycle-GAN model in such a way that the synthetic images generated by this new, improved model, called CSC-GAN, successfully augmented a tiny facade dataset and improved the performance of two

(a) (b) (c)

FIGURE 5.6: **Comparing images.** Real image (a), regular CycleGAN model (b) and CSC-GAN model (c). The labels were omitted.

semantic segmentation methods, *i.e.* DeeplabV3 [19] by approximately 4%, and the FCN [81] model by roughly 3%.

Chapter 6

Learning an Augmentation Strategy for Sparse Datasets

Data augmentation is a practical approach to increasing the amount and diversity of available data. Moreover, it aids in imparting the model with knowledge about the consistency in the data domain [28]. While extensive efforts have been made to develop improved network architectures [142, 44, 47, 52, 104, 116, 120], it has been demonstrated [66, 93] that data augmentation can also be efficiently utilised to incorporate data invariance and substantially enhance the generalisation capabilities of existing deep learning models. Unfortunately, the effectiveness of an augmentation technique on a specific dataset and application does not guarantee its success when applied to other datasets and applications [28]. Therefore, automated discovery of augmentation strategies from data is gaining traction as a promising approach to automate the augmentation process [29, 74, 28, 75, 71, 125].

In this chapter, we intend to answer research question 4 and reinforce the findings of the previous chapters regarding questions 2 and 3 (as outlined in section 1.2). In order to do this, we combine three methods to build a robust augmentation policy suitable for sparse datasets and applied to semantic segmentation tasks. Our strategy is based on SPADE [92], a spatially-adaptive normalisation model that synthesises photorealistic images given a semantic input layout, initially introduced in the previous chapter. As far as we know, we are the first to provide an augmentation technique based on SPADE, the current state-of-the-art label-to-image translation method, which can successfully increase the performance of a semantic segmentation model.

To assist the model in preserving semantic information while translating from label to image, we added our semantic loss introduced in the previous chapter, as shown in equation 5.4. Further improvement can be achieved when this robust method is connected with the self-attention adversarial network (SAGAN) [136], which enables attention-driven, long-range dependency modelling for image synthesis tasks. We show that our strategy learns an augmentation policy that regards the synthetic images as missing data points in the training set, as shown in Figure 6.1, where, in the first row, we show a t-SNE [83] plot on four datasets (in blue) and the augmentation data generated by our method (in red). In addition, in the second row, the relative improvement brought by our method, per class, when compared to the baseline, that is, the data set without augmentation.

With this strategy, we improve the generalisation performance of three state-of-the-art semantic segmentation networks: Criss-Cross Network (CCNet) [53], Fully-Convolutional Network (FCN) [110] and DeeplabV3 [19]. Our method leads to an improvement in the mean intersection over union (mIoU) score of between 0.5 and 14 percentage points, under different circumstances. This is a remarkable achievement if we consider that no new data or preliminary information has been provided to the model. In addition, CCNet, FCN and DeeplabV3 are already extremely good at extracting generalisable knowledge from any data set.

We prove our method's strength on four datasets: (1) The cityscape dataset [27], a street scenes dataset from 50 different cities, with 5k high-quality pixel-level annotations. (2) The SUN RGB-D dataset [118], a dataset with about 10k images that focus on scene understanding, with dense annotations in both 2D and 3D, for both objects and rooms. (3) A sparse building facade dataset [103] that consists of about 600 images. (4) A tiny grayscale dataset (Box) made only of boxes of packages arranged on pallets that consist of only about 400 images, relatively dense compared with the other datasets. In order to complete this task, we select images at random from the original datasets. We simulate sparsity from the Cityscape and SUN RGB-D datasets by artificially creating four datasets with different sizes: 0.5k, 1k, 2k and 4k. By simulating different sizes in the Cityscape and the SUN RGB-D datasets, we intend to demonstrate that sparsity is a factor that affects the performance of GAN-based data augmentation strategies and answer research question 4.

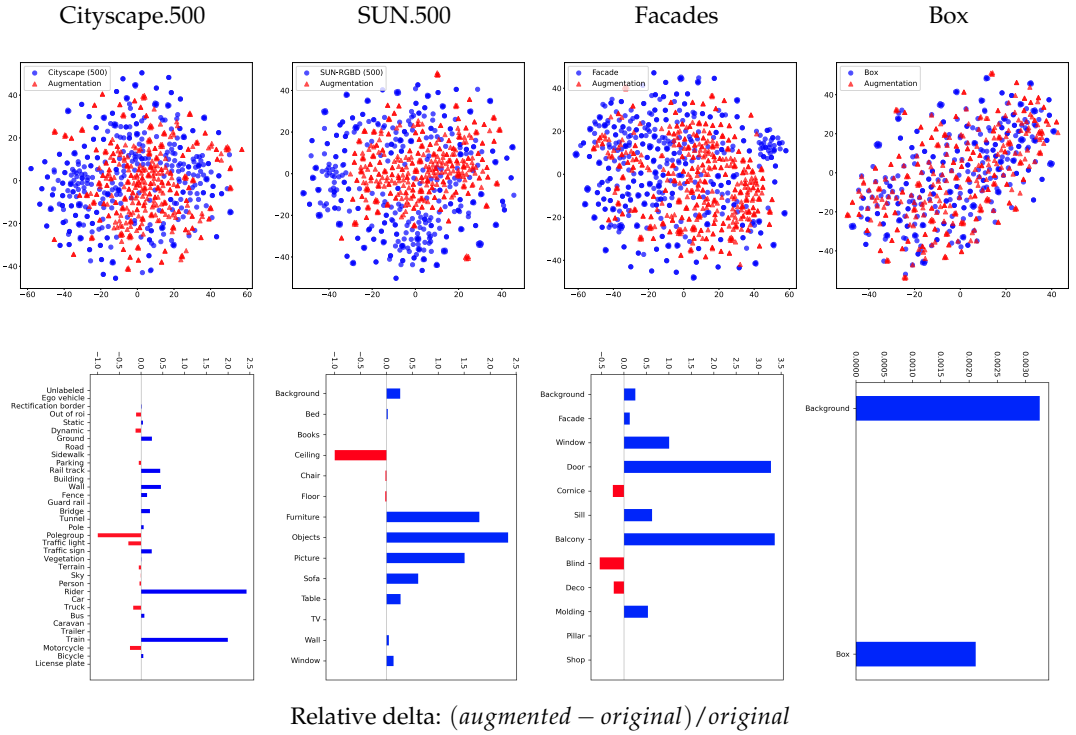


FIGURE 6.1: **Our method in action.** In the first row, A t-SNE [83] plot of the extracted VGG16's features [116] on four datasets and the augmentation data generated by our method. In the second row, the relative improvement brought by our method when compared with the baseline, *i.e.* the dataset without augmentation; the blue bars represent improved labels, while the red bars indicate that the augmented dataset performed worse than the baseline. Our method provides augmentation data that boost pixel-level semantic segmentation accuracy by filling the original dataset gaps.

Through our experiments, we verify that our proposed augmentation strategy provides an affordable and consistent pipeline suitable for sparse datasets and can be applied to increase pixel-level semantic segmentation models' precision by generating images that fill in the gaps in the training set.

6.1 Related Work

Human-designed data augmentation. One of the first applications of data augmentation for deep learning is present in LeNet-5 [69], where the authors apply CNNs on handwritten digits classification, employing the MNIST dataset. By artificially augmenting the training set with distorting samples, the error rate on the test set dropped from 0.95% to 0.7%. Now, the best-ranked models on the MNIST dataset extensively apply data augmentations like scale and rotation to improve model performance [128, 25].

The AlexNet [67] architecture developed for image classification broadly applied label-preserving transformations to increase the size of the training set. Data augmentation was used in their experiments to improve the training set by a factor of 2048. This increase in size was achieved by randomly cropping patches, flipping them horizontally and varying the intensity of the RGB channels using PCA colour augmentation. By this augmentation, the model error was reduced by over 1%.

Why a small error reduction can be relevant? In some cases, when dealing with massive amounts of data, even small accuracy gains can be significant. A 1% or less improvement could mean correctly classifying thousands or even millions of additional data points, leading to more reliable insights or predictions. Again, accuracy is usually associated with the confidence and trust users place in a system. Even a slight improvement in accuracy can enhance the perception of a model's reliability and effectiveness. Users may be more willing to adopt or rely on a system that demonstrates consistent incremental improvements, even if those improvements are relatively small. Furthermore, advances in machine learning and artificial intelligence are typically achieved through incremental progress. Many breakthroughs are the result of numerous small improvements rather than a single revolutionary change. A 1% gain in accuracy can be seen as a step forward in this iterative process, bringing us closer to better models and solutions.

Although artificially applying label-preserving transformations [115, 25, 26] is the most popular and straightforward method to overcome overfitting and improve model generalisation, they are manually designed and demand substantial domain knowledge to create suitable data transformations. In a study by Kexin *et al*., it was shown that CNNs can be misled using basic geometric transformations like rotation and translation [96]. Additionally, the same group demonstrated in another study that a self-driving car can be made to turn right and collide with a guardrail simply by slightly darkening the input image [95].

Typically carried out when a model is being trained, data augmentation can also be used at test time to improve accuracy [66, 86, 59]. Test-time augmentation (TTA) involves grouping predictions from multiple transformed versions of a given test input aiming to obtain a softened prediction. Its popularity is increasing because it is easy to practice as it does not require new data or changes in the underlying

model. Nevertheless, despite its popularity, there is relatively limited research on when and why TTA works, it has been shown recently [109] that even when TTA produces a net improvement in accuracy, it can turn many correct predictions into incorrect ones.

Machine-designed data augmentation. Deviating from human-designed augmentation strategies, applying machine-designed augmentation policies has arisen as a promising methodology to overcome the limitations of manually designing augmentation approaches. For instance, with *Smart Augmentation*, Lemley *et al*., instead of relying on hard-coded transformations, introduces a neural network that is trained to combine existing samples in order to generate additional data to augment the training set [71]. Lin *et al*. proposed a hyper-parameter learning method for auto-augmentation strategy, named *OHL-Auto-Aug*, which formulates the augmentation policy as a parameterised probability distribution, reporting a search cost for augmentation strategies that is up to 60x faster than *Smart Augmentation*, for achieving the same level of accuracy [76].

A new augmentation method based on style transfer that randomises textures, contrasts and colours during training is presented by Jackson *et al*. [57]. This randomisation level is accomplished by adopting an arbitrary-style transfer network to perform style randomisation by sampling target-style embeddings from a multivariate normal distribution instead of computing them from a style image. In addition, the authors found that their style augmentation method can enhance network performance by 8.5% when combined with a mix of seven traditional augmentation techniques. In the experimental results section, we compared our method to two state-of-the-art augmentation techniques and arrived at the same conclusion.

Cubuk *et al*. describe a procedure called *AutoAugment* that uses reinforcement learning to automatically find a data augmentation strategy when a target dataset and a model are provided [28]. Subsequently, the authors introduce what they claim to be an evolution of *AutoAugment* method, named *Fast AutoAugment*, that finds augmentation policies via a more efficient search strategy based on density matching [74]. Finally, the same authors propose the *RandAugment* algorithm that uses a grid search to optimise a hyperparameter that controls the magnitude applied to transformations, like translation and colour inversion, that are randomly sampled

[29]. The authors have reported an increase in accuracy of 0.6% over the previous state-of-the-art and a 1.0% improvement over the baseline augmentation for the ImageNet dataset.

GAN-based augmentation. Generative adversarial networks (GANs) [43], introduced by Goodfellow *et al*., have shown extraordinary results in many computer vision tasks such as style transfer [61, 130], image generation [16, 32, 6, 7], image translation [55, 140, 23], multi-domain image translation [23] and super-resolution [70]. This extraordinary performance has resulted in more attention on how they can be applied to the data augmentation task [117, 63, 4, 15, 108, 140, 141]. One of the domains where GANs have been applied successfully is for data augmentation in Medical Imaging. For instance, MedGAN [8] proposes a new architecture by joining the adversarial framework with a new combination of non-adversarial losses that successfully outperforms the current state-of-the-art in image translation in the medical area. The medical field has several GAN-based methods [68, 99] successfully applied to the segmentation task, and an in-depth review of these methods can be seen in [122], where Tajbakhsh *et al*. describe a multitude of methods based on CycleGAN [140] or conditional GAN [87] to synthesise from x-rays to user-defined images of injured skin.

Based on the SPADE model [92], Bargsten *et al*. intend to simulate realistic ultrasound speckles, introducing a speckles layer in the SPADE model generator, and also targeting dataset augmentation. They report that their SpeckleGAN model has little impact on performance when it comes to the segmentation task [10].

Our method also relies on GANs for data augmentation. We combine two recent GAN methods, SPADE [92] and Self-attention [136], aiming to build an automatically finding augmentation policy that is suitable for sparse datasets. Our method can enhance any dataset by creating an augmentation policy that considers synthetic images as absent data points in the training set. Unlike the other methods mentioned in this section, our approach is specifically designed for semantic segmentation and can generate images based on spatial labels. Spatial labels in semantic segmentation typically involve assigning labels to pixels or regions based on their visual appearance, context, and spatial relationships with neighbouring pixels.

6.2 Methodology

This section presents our framework that combines three methods in order to build a robust automatic finding augmentation policy, *i.e.* an augmentation strategy that is not manually designed, and that considers synthetic images as missing data points in the training set. Our approach enhances pixel-level semantic segmentation accuracy by generating additional data that fills in the gaps of the original dataset. Figure 6.1 is an example of our method in action. In the first row, we present a t-SNE [83] plot that displays four datasets (in blue) and the augmentation data produced by our approach (in red). Furthermore, in the second row, we showcase the improvement achieved by our method for each class, compared to the baseline, which is the dataset without augmentation. Refer to section 6.3 for the experimental results.

Figure 6.2 highlights our additions to the SPADE model described int section 6.2.1, which are summarised below:

1. An attention mechanism that complements the convolution process and supports modelling long-range, multi-level dependencies across image areas.

2. A semantic loss that helps to strengthen local and global structural consistency by evaluating how accurate the synthetically generated image is compared to a previously trained semantic segmentation model g.

By enhancing SPADE with these two aspects, we aim to improve the synthetic image quality, generate samples that belong to the same data distribution as the source data, and that also will be considered as a new sample by the semantic segmentation models, which is a fundamental condition for a successful augmentation method.

6.2.1 SPADE

To generate images for semantic segmentation, a specific geometry must be respected as the objects in the synthetic images must be generated following the objects' position in the labels. For example, in an image of a street scene, labels may include assigning *road* to pixels representing the road surface, *building* to pixels representing buildings, *car* to pixels representing cars, and so on. Each pixel or region is labelled

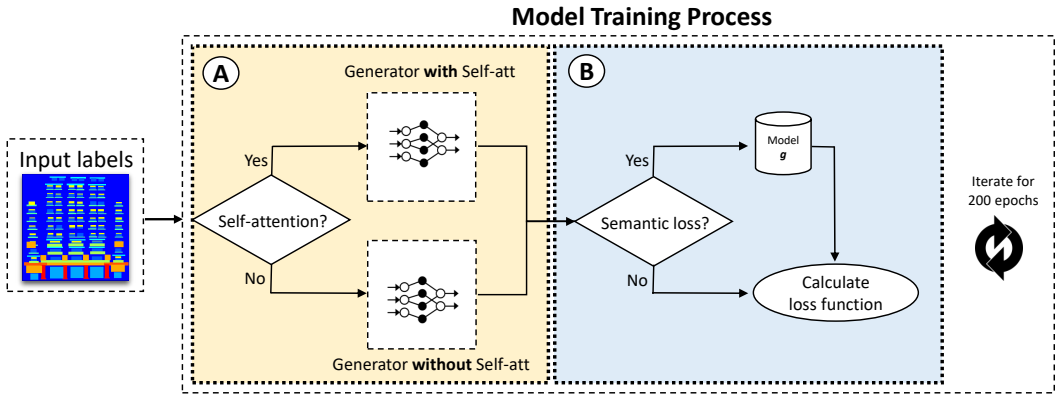


FIGURE 6.2: **The training process.** This image highlights the two contributions we made to the SPADE model targeting the synthetic image quality enhancement. **(A)** The self-attention mechanism and in **(B)** the semantic loss. Both can be turned on/off, and therefore we have four training possibilities: both off, which is the original SPADE model. Alternate on, and both on. That brings four different models. See Table 6.2 for the possibilities.

according to its semantic content, enabling the segmentation model to understand and differentiate different objects or regions within the image. A natural way of following this restriction is by using SPatially-Adaptive (DE)normalisation (SPADE) [92], a new conditional normalisation method for semantic image synthesis. SPADE layers transform segmentation masks into feature maps γ and β by feeding them through two convolutional layers. The generated parameters γ and β, which are tensors with spatial dimensions, are multiplied and added to the normalised input image x activation map, element-wise. Pixel values $x_{n,c,h,w}$ of input feature maps to be normalised are transformed as follows:

$$x'_{n,c,h,w} = \gamma_{c,h,w} \frac{x_{n,c,h,w} - \mu_c}{\sigma_c} + \beta_{c,h,w}, \tag{6.1}$$

where the indexes (n, c, h, w) refer to the batch size, the number of channels, the height and the input width. The parameters μ_c and σ_c denote the channel-wise mean and standard deviation of the input feature map x.

The generator employs many ResNet blocks [47] with upsampling layers, and since each residual block runs at a different scale, the semantic map is downsampled to match the resolution needed for learning the modularisation parameters γ and β. The discriminator does not apply SPADE and follows the pix2pixHD [129] discriminator, based on PatchGAN [56], that takes as input the concatenation of the segmentation map and the input image.

6.2.2 Semantic consistency loss

We strive to encourage high semantic consistency after image translation by the SPADE model. To enable this, we pre-train a semantic segmentation model g in the same *training* set used to train SPADE. As a second step, we use this pre-trained model g to assess the synthetic images' quality during SPADE training, as described in [4].

By fixing the model g weights during SPADE training, we ensure that an improvement in the segmentation results, obtained when the augmented data is present, is due to an increase in the synthetic image quality. Using this segmenter model g, we introduce the semantic consistency objective as

$$\mathcal{L}_{sem}(g, F) = \mathbb{E}_{y \sim p_{data}(y)} \mathcal{L}_{cs}[g(F(y)), y], \tag{6.2}$$

where $\mathcal{L}_{cs}[.,.]$ is the cross-entropy loss [51] matching two segmentation masks $g(F(y))$ and y. This equation implies that given a label $y \in Y$, the model g predicts the labels for $F(y)$, *i.e.*, the synthetic image produced using the real label y. The loss $\mathcal{L}_{sem}$ evaluates this prediction, and its result is added to the SPADE objective function, expecting to assist in improving the overall synthetic image quality.

6.2.3 Self-attention

The attention mechanism was first introduced as a tool for Neural Machine Translation (NMT) [9] in an encoder-decoder architecture intending to allow the decoder to automatically search for parts of the input sentence that are more relevant for the translation. Self-attention is an attention mechanism that relates different parts of a given input, aiming to model long-range dependencies on it. Cheng *et al*. [22] presents a recurrent network for machine reading that uses the self-attention mechanism to enhance the ability to discover relations among tokens. In the case of images, the self-attention tool is applied to better model the relationship among spatial regions in the input image. In SAGAN [136], the self-attention module complements the convolution process and supports modelling long-range, multi-level dependencies across image areas. The generator is then expected to synthesise images with

better details at each location, coordinating with details in distant parts of the generated image. This is accomplished by forking the convolutional image feature maps $\mathbf{x}$ out into three copies:

$$\begin{aligned} f(\mathbf{x}) &= \mathbf{W}_f \mathbf{x}, \\ g(\mathbf{x}) &= \mathbf{W}_g \mathbf{x}, \\ h(\mathbf{x}) &= \mathbf{W}_h \mathbf{x}. \end{aligned} \tag{6.3}$$

Then we apply the $softmax$ function to output the self-attention feature maps:

$$\begin{aligned} \beta_{ij} &= softmax(f(\mathbf{x}_i)^\intercal g(\mathbf{x}_j)), \\ \mathbf{o}_j &= \mathbf{W}_v(\sum_{i=1}^{N} \beta_{ij} h(\mathbf{x}_i)), \end{aligned} \tag{6.4}$$

where $\beta_{i,j}$ indicates the degree to which the model attends to the i^{th} position when synthesising the j^{th} region. $\mathbf{W}_f$, $\mathbf{W}_g$, $\mathbf{W}_h$ and $\mathbf{W}_v$ are all 1×1 convolution filters. The output $\mathbf{o}_j$ of the attention layer is the column vector $\mathbf{o} = (\mathbf{o}_1, \mathbf{o}_2, \cdots, \mathbf{o}_j, \cdots, \mathbf{o}_N)$.

Finally, the attention layer output is multiplied by a scale parameter and added back to the original input feature map:

$$\mathbf{y} = \mathbf{x}_i + \gamma \mathbf{o}_i, \tag{6.5}$$

where γ is a learnable scalar. We conducted experiments to address our problem and found that changing the γ value during training, as outlined in [92], did not significantly improve our method's overall performance. As a result, we kept the γ value constant at 1 for the entire training process.

6.3 Experimental Results

In this section, we will showcase the results of our approach on four different datasets: our Box dataset, Facade [103], Cityscape [27] and the SUN RGB-D [118] dataset.

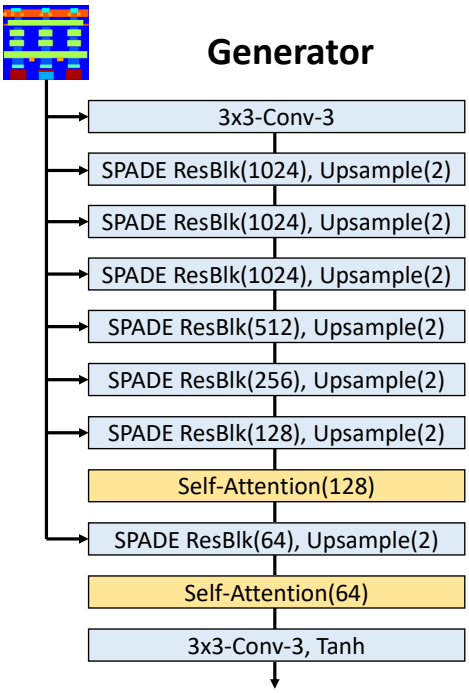


FIGURE 6.3: **The new generator.** Sketch of SPADE generator architecture after adding the self-attention module. It contains a sequence of SPADE residual blocks with upsampling layers. The last SPADE layer is between two Self-attention layers. The term *3x3-Conv-3* means a *3-by-3* convolutional layer with three convolutional filters, and *SPADE ResBlk* is the SPADE residual block. We made the network deterministic by starting with a downsampled segmentation map instead of random z.

6.3.1 Experimental Setup

To simulate sparseness on the Cityscape and SUN RGB-D datasets, we artificially derived four datasets by randomly removing images from them. All datasets we used on the experiments are listed in Table 6.1. To augment the datasets listed in Table 6.1, we used the SPADE [92] model, the state-of-the-art label-to-image translation model. To obtain further improvement, we enhance the SPADE model with the semantic reinforcement loss [4] and self-attention [136].

Our model generator, as shown in Figure 6.3, now includes a self-attention module in addition to the original SPADE model generator. It consists of a series of SPADE residual blocks with upsampling layers. Our contribution involves placing a SPADE layer between two self-attention layers, which are the last layers of the sequence. The term *3x3-Conv-3* refers to a convolutional layer with three filters that are each 3-by-3 in size, and *SPADE ResBlk* stands for SPADE residual block.

To prove the strength of our proposed framework, we apply it to the semantic

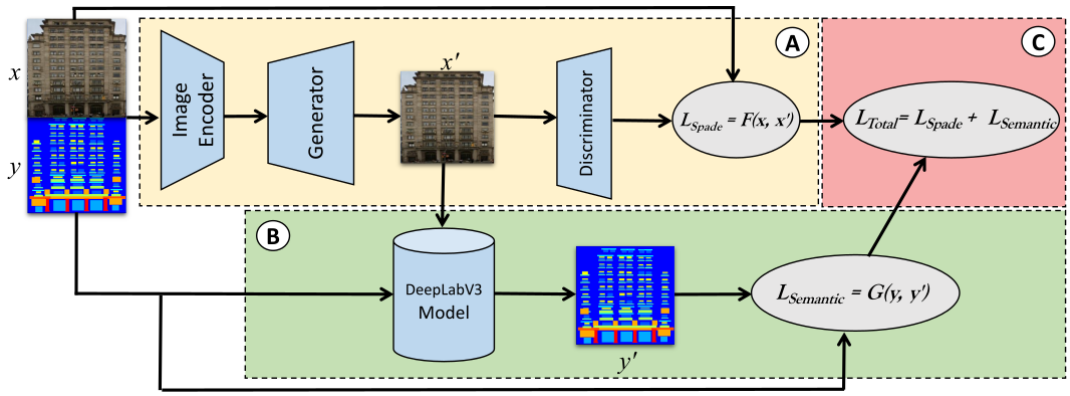


FIGURE 6.4: **Semantic loss two steps training.** First, we train a semantic segmentation model on the *training* set images x and labels y. Then we use it to assess the quality of the synthetic images through our model training. **(A)** We provide the source images x and labels y for the model and let it calculate the normal SPADE loss function. The image encoder makes the network deterministic by starting with a reduced resolution segmentation map of x instead of random z. **(B)** Using the synthetic image x' and the original label y as input to the semantic segmentation model, we generate the label y' and calculate the semantic loss. **(C)** Finally, we calculate the overall model loss.

Dataset Name	Size	Description
Box	438	Grayscale dataset gathered by the authors, made of boxes of packages arranged on pallets.
Facade	606	Base plus extended version of Facade dataset. [103]
City.500	500	500 randomly extracted images from Cityscape dataset. [27]
City.1000	1000	1000 randomly extracted images from Cityscape dataset.
City.2000	2000	2000 randomly extracted images from Cityscape dataset.
City.4000	4000	4000 randomly extracted images from Cityscape dataset.
SUN.500	500	500 randomly extracted images from SUN RGB-D dataset. [118]
SUN.1000	1000	1000 randomly extracted images from SUN RGB-D dataset.
SUN.2000	2000	2000 randomly extracted images from SUN RGB-D dataset.
SUN.4000	4000	4000 randomly extracted images from SUN RGB-D dataset.

TABLE 6.1: **Datasets.** Lists of all datasets used in our experiments. The split was 80% for training and 20% for the test. The test set is the same across all experiments.

segmentation problem, and to access its performance, for each dataset, we trained two semantic segmentation models: Fully-Convolutional Network (FCN) [110], and DeeplabV3 [20], both models relying on a ResNet101 [47] backbone. Each model is trained for 50 epochs with a learning rate set to 0.0002, and the Adam optimiser [64]. For the Cityscape dataset, we also trained the Criss-Cross Network (CCNet) [53] model. CCNet is a new state-of-the-art semantic segmentation model that uses an attention mechanism to map long-range dependencies and capture useful contextual information by collecting the surrounding pixels on a criss-cross path through a novel criss-cross attention module. We trained the CCNet model using an initial learning rate set to 0.01, weight-decay set to 0.0005 and the model was trained for 60000 steps. To enable training on a relatively low-end, single-GPU architecture, we resized the images in the Cityscape dataset to 512×256, and all experiments in this

Augmentation Type	Description
SPADE	Augmentation based only on SPADE synthetic images.
SPADE+self-att	SPADE model enhanced with the self-attention mechanism. See section 6.2.3.
SPADE+seg-loss	SPADE model enhanced with the segmentation loss. See section 6.2.2.
SPADE+seg-loss+self-att	SPADE model enhanced with both self-attention mechanism and the segmentation loss.

TABLE 6.2: **Augmentations.** We experiment with four types of augmentation in order to evaluate the benefit of each of the components of our proposed method.

dataset were performed with these resized images. All our models were trained using a single GPU GeForce RTX 2080 Ti with 11GB of memory. The amount of time needed for training varies depending on the dataset. The Box dataset takes approximately 2 hours to train, while the Cityscape.4000 dataset takes around 48 hours.

Each dataset has been split into a *training* X_S set, with 80% and a *test* X_T set, with 20%. As shown in Table 6.2, there are 4 augmentation variations of the *training* set per dataset in order to evaluate the various components of our proposed method. The augmentation is done by the following procedure: For each label $y \in X_S$, a synthetic image $x = F(y)$ is generated and the pair (y, x) is added to the dataset. By the end, a new *training* set $X_S^{'}$ is created with a size twice as X_S. This is the consequence of adding one synthetic image per each label $y \in X_S$ to the new *training* set $X_S^{'}$.

All models' performances are reported using the mean intersection-over-union (mIoU) score [111] and pixel accuracy, evaluated over the *test* set, which is constant across all experiments.

When the semantic reinforcement loss is used, two steps are needed: First, we train the semantic segmentation model g on the regular dataset, without any augmentation, which is used later as a labeller during the SPADE training. Then we train SPADE with the semantic reinforcement loss, as described in 6.2.2.

In Figure 6.4, we outline the two-step training process. The initial step involves training a semantic segmentation model on the *training* set images x and labels y. Then we use it to assess the quality of the synthetic images through our model training. **(A)** We provide the source images x and labels y for the model and let it calculate the normal SPADE loss function. The image encoder makes the network deterministic by starting with a reduced resolution segmentation map of x instead of random z. **(B)** Using the synthetic image x' and the original label y as input to the semantic segmentation model, we generate the label y' and calculate the semantic loss. **(C)**

Finally, we calculate the overall model loss.

To confirm the benefits of our method, we also compared its effects on two augmentation strategies, a mix of three traditional augmentation techniques and the *RandAugmentation* [29] algorithm, which applies transformations that are randomly sampled. The traditional (Trad) augmentation methods are: `ShiftScaleRotate`, `RGBShift` and `RandomBrightnessContrast` [17]. For the *RandAugmentation* algorithm (Rand), we use the parameters $M = 5$ and $N = 3$ as suggested in the paper and sample the transformations between all spatial level transformations in the **Albumentation** library [17] that simultaneously alter the input image and the label. We run these experiments on the DeepLabV3 model using the same setup as the previous experiments.

6.3.2 Evaluation

Tables 6.3, 6.4 and 6.5 shows the results. Table 6.3 contrasts the Facade and the Box dataset; Table 6.4 shows the results for the SUN RGB-D datasets, and Table 6.5 shows the results for the Cityscape datasets. For Tables 6.3 and 6.4, each experiment was performed five times and the results listed are the mean and the standard deviation for these five executions.

In Table 6.5, for the CCNet model, we report only the mIoU, which is the metric reported by the model source code provided by the authors [1]. In addition, the mean and standard deviation reported are for the five runs of the model. The mean and standard deviation reported for DeeplabV3 and FCN models are calculated for the five runs of the model, as in Table 6.3 and Table 6.4. Examples of synthetic images generated by our method are displayed in Figure 6.9.

The given results show that the dataset augmented with images from the SPADE model, enhanced with the semantic reinforcement loss and the self-attention mechanism, delivers the best results. These results are statistically distinct from the baseline (2-tailed t-test, 5% significance level), and are shown in bold in the results tables.

As stated before, our method performs better on a more sparse dataset, as Figure 6.6a demonstrates. This figure compares the gain in the mIoU [111] score, for the SUN RGB-D and Cityscape datasets, against dataset size. It shows the percentage

[1] `https://github.com/speedinghzl/CCNet`

		DeeplabV3 [19]		**FCN** [110]	
Dataset name	**FID**	**mIoU**	**Accu.**	**mIou**	**Accu.**
Box		0.875 (0.002)	0.929 (0.001)	0.875 (0.003)	0.929 (0.002)
Box+SPADE	160.451	0.875 (0.002)	0.929 (0.002)	0.874 (0.003)	0.928 (0.002)
Box+SPADE+self-att	151.186	0.875 (0.003)	0.929 (0.002)	0.878 (0.003)	**0.931 (0.002)**
Box+SPADE+seg-loss	**150.800**	0.877 (0.001)	**0.931 (0.001)**	0.876 (0.003)	0.930 (0.002)
Box+SPADE+self-att+seg-loss	151.697	**0.879 (0.005)**	0.931 (0.003)	**0.880 (0.003)**	**0.931 (0.002)**
Facades		0.552 (0.016)	0.706 (0.013)	0.558 (0.003)	0.711 (0.002)
Facades+SPADE	85.512	0.565 (0.009)	0.716 (0.008)	0.572 (0.004)	0.723 (0.003)
Facades+SPADE+self-att	91.302	0.583 (0.002)	0.731 (0.002)	0.580 (0.003)	0.729 (0.003)
Facades+SPADE+seg-loss	91.455	0.571 (0.004)	0.722 (0.003)	0.576 (0.003)	0.725 (0.002)
Facades+SPADE+self-att+seg-loss	**85.371**	**0.586 (0.002)**	**0.734 (0.002)**	**0.589 (0.002)**	**0.736 (0.001)**

TABLE 6.3: **Experimental results**. Accu. is the pixel accuracy; the FID [49] metric is calculated against the dataset without augmentation. The mIoU and the pixel accuracy results reported are the mean and the standard deviation after 5 executions of the experiment. The test set used in all experiments is the same used when no augmentation is present.

improvement between the best mIoU average, in bold at Table 6.4 and Table 6.5, and the baseline average, the first row on each table section. It ranges from an improvement of 6.2% to 14.2% for the SUN RGB-D dataset and from 1.2% to 4.1% for the Cityscape dataset.

To support the density hypothesis, we chose to calculate our datasets density using the K-nearest neighbour algorithm [2]. To that end, for each image $\mathcal{I} \in \mathcal{S}$, we selected the median distance $Med(\mathcal{X})$, among the ordered list $\mathcal{X}$ of all $K = 5$ nearest neighbours of $\mathcal{I}$, and calculated the mean μ over all dataset images median distance $Med(\mathcal{X})$.

$$Med(\mathcal{X}) = \mathcal{X}\,[3]\,, \tag{6.6}$$

$$\mu = \frac{1}{N}\sum_{n=1}^{N} Med(\mathcal{X}_i), \tag{6.7}$$

where N is the dataset $\mathcal{S}$ size. We take μ as our dataset density metric. We calculated the density of the dataset using the directly extracted VGG16 features and the 2D t-SNE points calculated on these features. Although the scale of values between the two calculations is different, their results agree with each other, as shown in Table 6.6. Figure 6.6b shows the relationship between the dataset density and the performance gain for the SUN RGB-D and Cityscape datasets, respectively. This

[2] `https://scikit-learn.org/stable/modules/neighbors.html#unsupervised-neighbors`

		DeeplabV3 [19]		**FCN** [110]	
Dataset name	**FID**	**mIoU**	**Accu.**	**mIou**	**Accu.**
SUN.500		0.360 (0.021)	0.515 (0.022)	0.379 (0.009)	0.538 (0.008)
SUN.500+SPADE	**119.767**	0.403 (0.012)	0.560 (0.012)	0.404 (0.022)	0.561 (0.022)
SUN.500+SPADE+self-att	132.359	0.407 (0.006)	0.565 (0.006)	0.423 (0.008)	0.580 (0.008)
SUN.500+SPADE+seg-loss	123.174	0.410 (0.004)	0.568 (0.004)	0.425 (0.011)	0.583 (0.011)
SUN.500+SPADE+self-att+seg-loss	135.961	**0.411 (0.006)**	**0.569 (0.006)**	**0.431 (0.008)**	**0.588 (0.008)**
SUN.1000		0.357 (0.006)	0.508 (0.006)	0.383 (0.005)	0.537 (0.006)
SUN.1000+SPADE	110.611	0.378 (0.014)	0.529 (0.014)	0.393 (0.016)	0.529 (0.014)
SUN.1000+SPADE+self-att	111.069	0.392 (0.003)	0.544 (0.003)	0.394 (0.008)	0.547 (0.008)
SUN.1000+SPADE+seg-loss	101.117	0.389 (0.006)	0.541 (0.006)	0.396 (0.012)	0.549 (0.012)
SUN.1000+SPADE+self-att+seg-loss	**98.200**	**0.393 (0.003)**	**0.545 (0.003)**	**0.405 (0.008)**	**0.558 (0.008)**
SUN.2000		0.396 (0.006)	0.552 (0.005)	0.405 (0.019)	0.561 (0.019)
SUN.2000+SPADE	79.101	0.425 (0.009)	0.580 (0.009)	0.432 (0.009)	0.588 (0.009)
SUN.2000+SPADE+self-att	75.115	0.418 (0.006)	0.574 (0.006)	0.437 (0.003)	0.593 (0.003)
SUN.2000+SPADE+seg-loss	**69.400**	0.422 (0.008)	0.578 (0.007)	0.434 (0.008)	0.589 (0.008)
SUN.2000+SPADE+self-att+seg-loss	81.646	**0.427 (0.007)**	**0.582 (0.007)**	**0.441 (0.010)**	**0.596 (0.010)**
SUN.4000		0.420 (0.005)	0.575 (0.005)	0.434 (0.009)	0.588 (0.009)
SUN.4000+SPADE	62.062	0.441 (0.009)	0.595 (0.008)	0.450 (0.009)	0.604 (0.009)
SUN.4000+SPADE+self-att	56.956	0.437 (0.011)	0.591 (0.010)	0.448 (0.010)	0.602 (0.009)
SUN.4000+SPADE+seg-loss	68.293	**0.446 (0.006)**	**0.599 (0.005)**	0.442 (0.006)	0.596 (0.006)
SUN.4000+SPADE+seg-loss+self-att	**52.882**	0.436 (0.011)	0.591 (0.010)	**0.456 (0.003)**	**0.610 (0.002)**

TABLE 6.4: **Experimental results**. Accu. is the pixel accuracy; the FID [49] metric is calculated against the dataset without augmentation. The mIoU and the pixel accuracy results reported are the mean and the standard deviation after 5 executions of the experiment. Four datasets were randomly extracted from the original SUN RGB-D dataset: one with 500, 1000, 2000 and 4000 images. The test set used in all experiments is the same used when no augmentation is present.

		CCNet [53]	**DeeplabV3** [19]		**FCN** [110]	
Dataset name	**FID**	**mIoU**	**mIoU**	**Accu.**	**mIoU**	**Accu.**
City.500		0.436 (0.004)	0.772 (0.001)	0.861 (0.001)	0.774 (0.006)	0.871 (0.004)
City.500+SPADE	78.464	0.452 (0.003)	0.782 (0.002)	0.876 (0.001)	0.778 (0.006)	0.872 (0.003)
City.500+SPADE+self-att	**74.401**	0.453 (0.002)	0.781 (0.002)	0.876 (0.001)	0.773 (0.007)	0.871 (0.005)
City.500+SPADE+seg-loss	75.485	0.453 (0.003)	0.780 (0.003)	0.875 (0.002)	0.776 (0.004)	0.872 (0.003)
City.500+SPADE+self-att+seg-loss	74.897	**0.455 (0.003)**	**0.785 (0.001)**	**0.879 (0.001)**	**0.782 (0.002)**	**0.876 (0.001)**
City.1000		0.495 (0.004)	0.779 (0.001)	0.875 (0.001)	0.773 (0.004)	0.871 (0.002)
City.1000+SPADE	55.981	0.495 (0.002)	0.784 (0.000)	0.878 (0.000)	0.776 (0.001)	0.872 (0.000)
City.1000+SPADE+self-att	**50.039**	0.510 (0.003)	0.784 (0.001)	0.878 (0.002)	0.778 (0.004)	0.874 (0.002)
City.1000+SPADE+seg-loss	52.227	0.501 (0.000)	0.784 (0.002)	0.878 (0.002)	**0.783 (0.002)**	**0.877 (0.001)**
City.1000+SPADE+self-att+seg-loss	51.183	**0.520 (0.001)**	**0.786 (0.001)**	**0.879 (0.001)**	0.781 (0.001)	0.876 (0.001)
City.2000		0.583 (0.001)	0.790 (0.001)	0.877 (0.001)	0.790 (0.000)	0.882 (0.001)
City.2000+SPADE	47.221	0.583 (0.001)	0.792 (0.000)	0.883 (0.000)	0.789 (0.001)	0.878 (0.001)
City.2000+SPADE+self-att	46.656	**0.595 (0.000)**	**0.793 (0.001)**	**0.885 (0.001)**	**0.793 (0.001)**	**0.883 (0.000)**
City.2000+SPADE+seg-loss	46.605	0.588 (0.002)	0.792 (0.001)	0.883 (0.001)	0.791 (0.000)	0.883 (0.000)
City.2000+SPADE+self-att+seg-loss	**45.892**	0.581 (0.003)	0.792 (0.000)	0.883 (0.000)	0.792 (0.001)	0.883 (0.000)
City.4000		0.589 (0.004)	0.761 (0.005)	0.856 (0.004)	**0.745 (0.001)**	0.844 (0.000)
City.4000+SPADE	52.442	0.592 (0.003)	0.760 (0.002)	0.854 (0.002)	0.742 (0.003)	0.841 (0.003)
City.4000+SPADE+self-att	**41.658**	0.586 (0.003)	**0.765 (0.002)**	**0.868 (0.001)**	0.738 (0.002)	0.840 (0.001)
City.4000+SPADE+seg-loss	53.585	**0.596 (0.003)**	0.764 (0.004)	0.858 (0.002)	0.743 (0.006)	**0.846 (0.004)**
City.4000+SPADE+seg-loss+self-att	48.459	0.591 (0.002)	0.762 (0.004)	0.857 (0.004)	0.743 (0.004)	0.844 (0.003)

TABLE 6.5: **Experimental results**. Accu. is the pixel accuracy; the FID [49] metric is calculated against the dataset without augmentation. The mIoU and the pixel accuracy results reported are the mean and the standard deviation after 5 executions of the experiment. Four datasets were randomly extracted from the original Cityscape dataset: one with 500, 1000, 2000 and 4000 images. The test set used in all experiments is the same used when no augmentation is present.

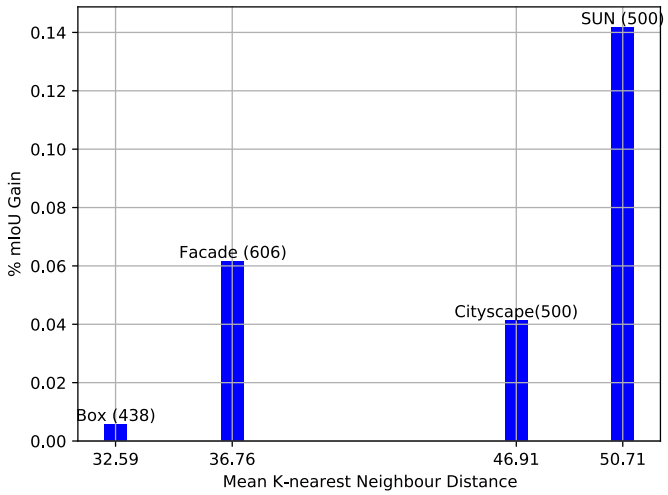


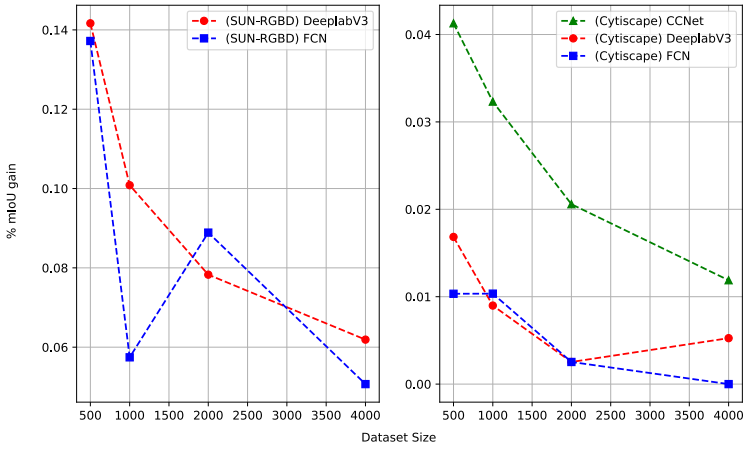
FIGURE 6.5: **Dataset Density.** Although the Box dataset is the smallest, it is also the densest dataset among the smallest. Therefore, its main gain was also the poorest.

shows that the lower the density, the greater the improvement brought to our GAN-based data augmentation method.

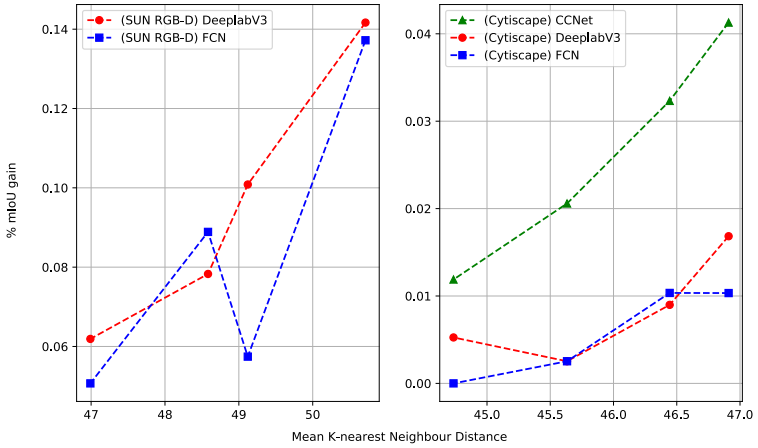
Not shown in Figure 6.6, but also relevant is the gain of 6.2% and 5.5% for the Facade dataset with the DeeplabV3 [19] and FCN [110] models, respectively. In [4], where the authors also experiment with the Facade dataset, the gain reported is of 4% with DeeplabV3 and 3% with the FCN model. As previously mentioned, machine learning and artificial intelligence make progress through incremental advancements rather than sudden breakthroughs. Even a minor improvement in accuracy is a step forward in this iterative process, leading us closer to improved models and solutions.

The worst result was achieved on the Box dataset, which improved only about 0.5% and 0.6% with DeeplabV3 and FCN, respectively. This result aligns with our expectation, as the Box dataset is the densest dataset among all datasets experimented with, even though it also is the smaller dataset. A comparison among the Box, Facade, City.500 and SUN.500 is shown in Figure 6.5.

Furthermore, in the first column at Tables 6.3, 6.4 and 6.5, the FID [49] metric reinforces our method's strength since for 4 out of 10 datasets present in the tables, our approach provided a better score when compared with the baseline, *i.e.* the dataset with no augmentation. For the SUN.2000 and the Box datasets, the better



(A)



(B)

FIGURE 6.6: **Performance comparison.** The relative improvement $(\frac{best_{mIoU}}{baseline_{mIoU}} - 1)$ is shown for each SUN RGB-D and Cityscape dataset size (a) and density (b) tested and for CCNet, DeeplabV3 and FCN. The $best_{mIoU}$ is computed as the average of the all $best_{mIoU}$ throughout all algorithm runs. Moreover, for all dataset sizes, a statistical pair-wise t-test indicates the performance of the CCNet, DeeplabV3 and FCN outperforms the baseline performance, at a 5% significance level.

		t-SNE 2D		VGG features	
Dataset	**Size**	**Mean Distance**	**Std**	**Mean Distance**	**Std**
Facades	606	2.387	0.063	36.762	1.269
Box	438	2.019	0.057	32.593	0.877
SUN	500	2.386	0.128	50.714	2.298
SUN	1000	2.241	0.082	49.121	1.714
SUN	2000	2.145	0.044	48.583	1.214
SUN	4000	1.915	0.062	46.990	0.633
Cityscape	500	2.410	0.047	46.929	0.057
Cityscape	1000	2.403	0.072	46.443	0.090
Cityscape	2000	2.325	0.127	45.632	0.039
Cityscape	4000	2.274	0.144	44.735	0.026

TABLE 6.6: **Dataset density**. The K-nearest neighbour's average distance to the dataset using the directly extracted VGG16 features and the 2D t-SNE points calculated on these features. As the data set gets smaller, the sparse it tends to be. We report the mean distance and the standard deviation calculated over 5 executions of the algorithm.

FID is also given by the SPADE model plus the semantic reinforcement loss. Moreover, for the Cityscape datasets, City.500 and City.4000, the better FID is given by the SPADE model augmented by the self-attention module. Only for City.1000 in Table 6.5, SPADE alone provided the better FID. The FID [49] score represents how close synthetic data are to the original, where the lower the value of FID, the closer these are.

It's important to note, as discussed in Chapter 5, that a lower FID value, which indicates a higher similarity to real images, doesn't necessarily result in better mIoU performance. Figure 6.7 demonstrates that, except for the SUN.400 dataset at the bottom, there isn't a significant correlation between FID and mIoU metrics. In simpler terms, just because the synthetic images have good visual quality, as measured by FID, doesn't mean they'll be effective in enhancing segmentation performance, as measured by mIoU.

Based on this information, it seems that while FID can offer useful information on the quality of synthetic images, it shouldn't be the only factor taken into consideration when determining whether they are appropriate for use in tasks such as data augmentation.

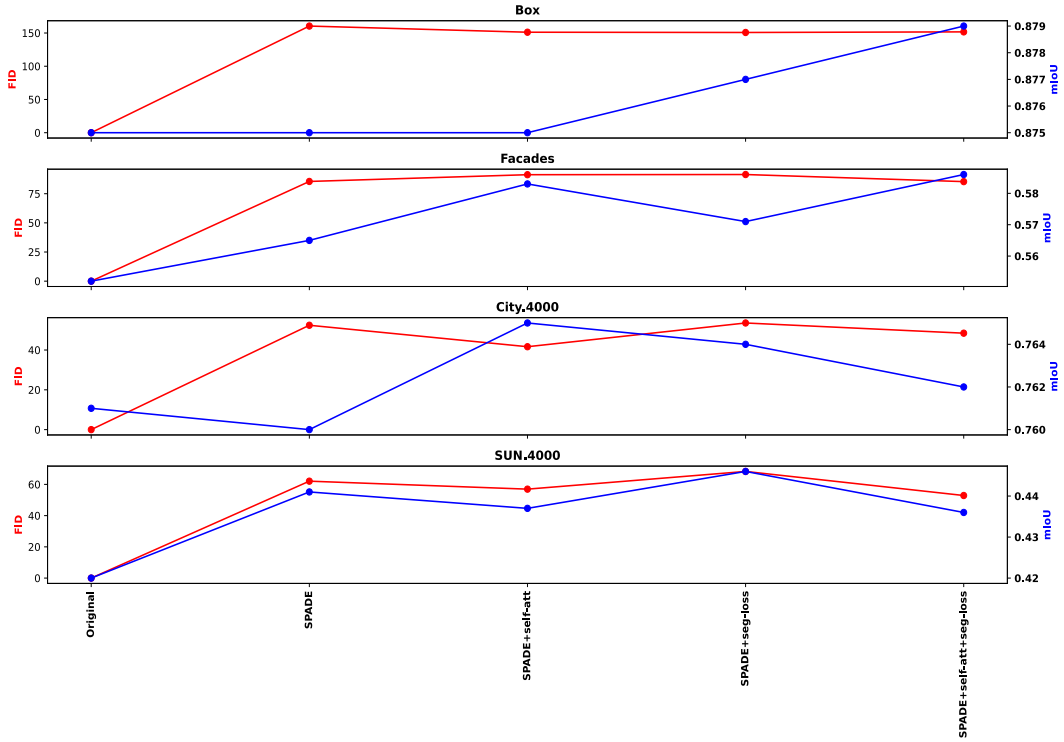


FIGURE 6.7: **FID x MioU.** Our experiments show that a lower FID score may not necessarily mean that the artificial image is better for data expansion purposes. Just because the synthetic images have good visual quality, as measured by FID, doesn't mean they'll be effective in enhancing segmentation performance, as measured by mIoU.

6.3.3 Regular Data Augmentation

Table 6.7 and Figure 6.8 summarises the results. The last two columns in Table 6.7 present the results when our method is associated with the traditional (Trad) augmentation strategy and the *RandAugmentation* (Rand) algorithm. As shown in the table, only for the SUN dataset did the traditional and *RandAugmentation* algorithm performed better than our (Ours) method. Furthermore, our strategy was superior for the Box, Facade, and Cityscape datasets. This points to the strength of our augmentation technique and the invariance it introduces into the model. However, our strategy also provided additional improvements when combined with the two regular methods across all datasets, as the last two columns in the table emphasise.

The key point to extract from the results obtained from these comparisons and the association of augmentation methods is to emphasise how difficult it is to develop an augmentation strategy manually. Although training a GAN method to generate synthetic images for data augmentation purposes can be expensive, as the cost of training a GAN can vary depending on the size and complexity of the dataset,

	Augmentation Approach					
Dataset	**None**	**Ours**	**Trad**	**Rand**	**Trad+Ours**	**Rand+Ours**
Box	0.875	**0.879**	0.847	0.860	0.826	0.868
Facades	0.552	**0.586**	0.531	0.570	0.536	0.577
SUN.500	0.360	0.411	0.409	0.418	**0.441**	0.440
SUN.1000	0.357	0.393	0.399	0.399	0.413	**0.414**
SUN.2000	0.396	0.427	0.433	0.428	**0.442**	0.438
SUN.4000	0.420	0.436	0.446	0.447	0.462	**0.464**
City.500	0.772	**0.785**	0.766	0.780	0.769	0.778
City.1000	0.779	**0.786**	0.769	0.777	0.776	0.782
City.2000	0.790	**0.792**	0.775	0.786	0.785	0.787
City.4000	0.761	**0.762**	0.634	0.700	0.635	0.697

TABLE 6.7: **Augmentation methods comparison.**. The reported mIoU is the mean after five runs of the experiment, and the standard deviation, not shown in this table, is always less than 0.01.

we believe we have demonstrated that it pays off.

6.3.4 Discussion

It is not immediately apparent why a low data density must correlate with the improvement brought about by an augmentation algorithm based on GAN, as proposed in this chapter. We hypothesise that when the dataset is small, such as Facade, Box, City.500 and SUN.500, it has gaps that the model can learn to fill. Consequently, the increased data is more effective in bringing about performance improvements. Therefore, the results presented here are in line with our assumption that considers synthetic images as missing data points in the training set: smaller datasets, in general, are better suited to augment with synthetic data than larger datasets because larger datasets also are denser and have fewer gaps to fill. Figure 6.10 illustrates this assumption, it is shown that as the dataset increases in size from top to bottom, it becomes denser, making it increasingly difficult to find a gap that can be filled with synthetic data.

6.4 Summary

This chapter proposes a fully automated data augmentation strategy that is suitable for sparse datasets. Using GAN-derived synthetic images, we demonstrated that our framework could improve results across three semantic segmentation tasks.

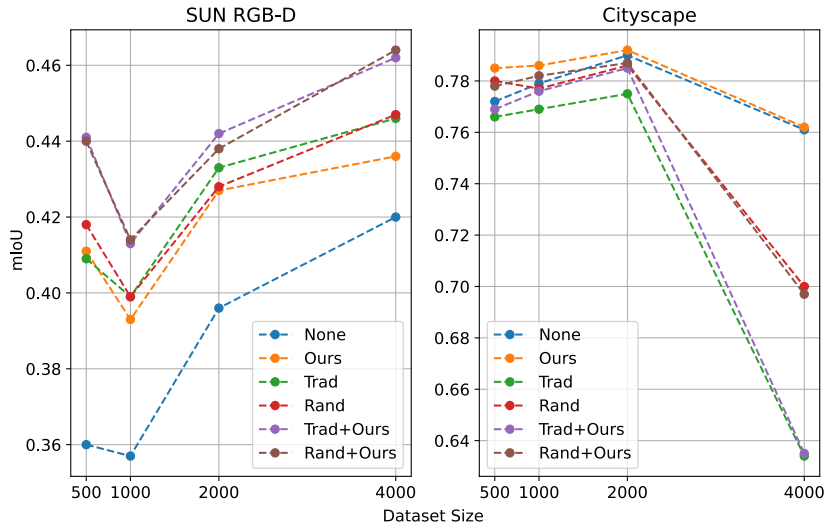


FIGURE 6.8: **Augmentation methods comparison.** We are comparing the mIoU improvement in the SUN-RGB-D and Cityscape datasets. For the SUN dataset, our method brought additional improvements when paired with traditional augmentation methods (Trad) and the *RandAugmentation* (Rand) algorithm. On the other hand, for the Cityscape dataset, our method is superior as a standalone technique. These charts emphasise how difficult it is to design a working augmentation strategy manually.

The framework has been shown to work best in limited data case scenarios and requires little overhead. All experiments were carried out on just a single GeForce RTX 2080 Ti with 11GB memory. The time required for training varies depending on the dataset used. For instance, it takes about 2 hours to train on the Box dataset, which is the smallest, and approximately 48 hours to train on the Cityscape.4000 dataset, which is the largest.

We also strive to demonstrate how a low data density situation can be correlated with the improvement brought by GAN-based augmentation algorithms: When the dataset is sparse, it has gaps that the model can learn to fill. Consequently, augmented data is more effective in bringing performance improvements. Our proposed method can provide an effective way to fill in these training data gaps and provide synthetic data that effectively bring performance improvement, leading to an improved mean intersection-over-union (mIoU) score of between 0.5 and 14 percentage points.

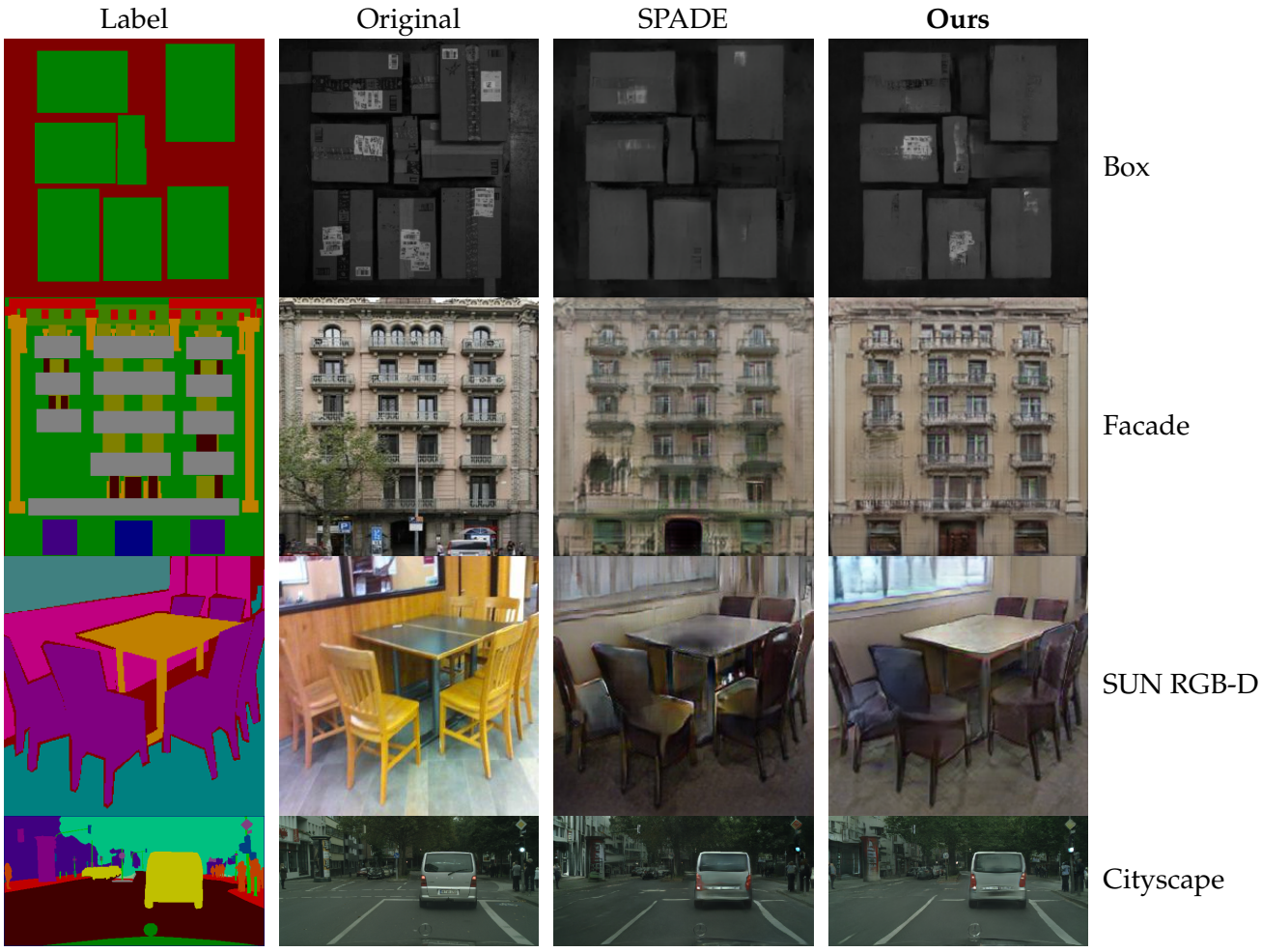


FIGURE 6.9: **Examples.** Samples synthesised by our SPADE-based method, with semantic loss and self-attention, are in the last column. Samples synthesised by SPADE are in column three. Across all datasets, the SPADE model enhanced with our semantic reinforcement loss function and self-attention method scored a better FID than SPADE alone.

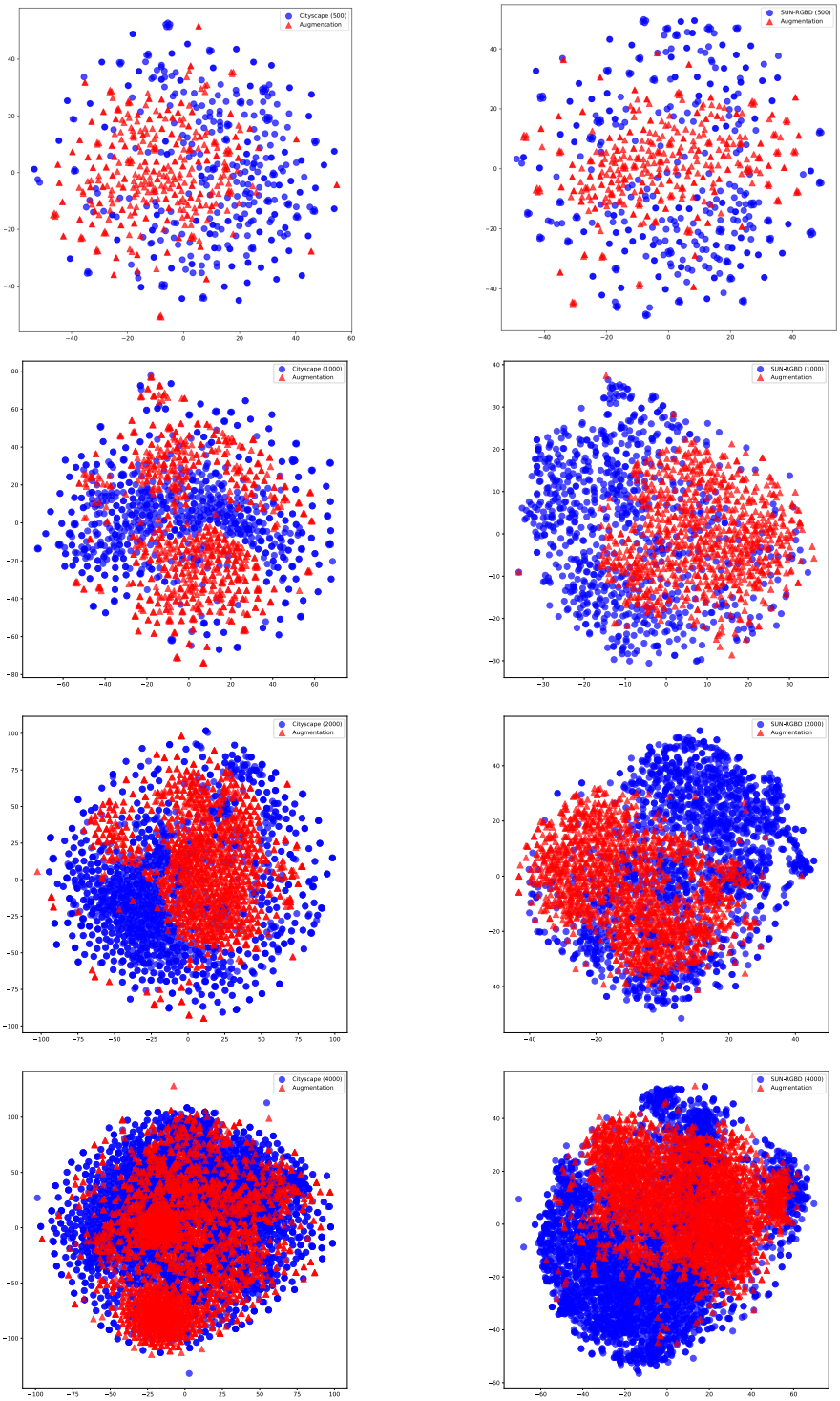


FIGURE 6.10: **Dataset density vs augmentation.** As the dataset grows in size, it becomes denser and, therefore, more complicated is finding a gap to fill with synthetic data. Images generated with t-SNE [83] on VGG16 [116] features. The first column is for the Cityscape dataset, and the second one is for the SUN RGB-D dataset. Each row is for a dataset artificially generated by randomly extracting images from the original dataset. We create four datasets by randomly extracting images from the original dataset: 500, 1000, 2000, 4000.

Part IV

Conclusions and Final Remarks

Chapter 7

Conclusions

In this dissertation, we studied deep generative models as an augmentation process for tiny datasets. In pursuit of this knowledge, we developed various methods along the way: (1) an image synthesis process, which artificially generates images containing some particular requested content, was created in Chapter 3. (2) another generative method that is capable of synthesising arbitrary-sized, high-resolution images derived from a single reference image was introduced in Chapter 4. Unlike the previous method in Chapter 3, which required three GANs, this new method only requires one GAN to synthesise images. (3) Building on the knowledge in the first part of this dissertation, we introduced in Chapter 5 a new loss function that allowed further improvements in the quality of the synthetic images so that they could be used successfully to augment a small facade dataset. (4) We demonstrate in Chapter 6, using various datasets, that additional improvement can be added when a self-attention mechanism is used in conjunction with a semantic loss function. (5) Finally, in Chapter 6, we also studied the influence of the density of a dataset on the augmentation process. This chapter concludes this dissertation by presenting our main contributions and directions for future research.

7.1 Contributions

We summarise the main contributions of this dissertation as follows;

7.1.1 Image Synthesis For Texture Generation

In Chapter 3, we address research question 1 (outline in section 1.1). We investigate the image synthesis process, which artificially generates images containing some particular requested content. As a result, we presented a deep generative model based on conditional GAN (cGAN) [87] that synthesises images based on a single source image. This method, named QuiltGAN, is aimed to generate texture and is inspired by the Image Quilting algorithm proposed by Efros *et al*. [34].

We have compared our method with IQ and IQ+GAN using the SIFID [107] metric in a few images. In two out of three comparisons, our method performed better. However, our method is expensive because it requires training three GANs to produce a synthetic image. As a result, we have decided to switch to a simpler method, which we introduced in Chapter 4.

7.1.2 Arbitrary Size Image Generation

In Chapter 4, we continue addressing research question 1 (outline in section 1.1) and investigate whether GANs [43], one of the most successful parametric models, might be employed to generate new, high-resolution images based on an individual reference image. We introduced *rcGAN*, a generative method capable of synthesising arbitrary-sized, high-resolution images derived from a single reference image used to train our model. The *rcGAN* framework is a two-step method that uses a randomly conditioned Generative Adversarial Network (rcGAN) trained on patches obtained from the reference image.

We compare the results of our two-steps methods with *SinGAN* [107], *InGAN* [112] and *Spatial-GAN* [58]. We acknowledge that our approach may not always produce the best quantitative results. However, we are convinced that our method is still superior to the other methods discussed in this context. This is because we maintain the original distribution while increasing variability, and also provide manual control over the generated synthetic image by allowing the selection of the seed.

7.1.3 Semantic Loss Constrained By Semantic Segmentation

Founded on our previous experience of using GANs for image synthesis, in Chapter 5, we extended the work presented in Part II by addressing research questions 2 and 3 (outline in section 1.1) and investigated to what extent an image-to-image translation model like CycleGAN [140] can be used as a data augmentation method. As a result, we proposed a new Cycle and Semantic Consistency (CSC-GAN) model, an adversarially trained image-to-image translation method that strengthens local and global structural consistency through cycle consistency and a new loss function that is constrained by semantic segmentation. In addition, our formulation encourages the model to preserve semantic information during the translation process.

This refinement enables using the proposed method for data augmentation in the semantic segmentation domain. The enhanced model successfully augmented a tiny facade dataset and improved the performance of two semantic segmentation methods, *i.e*. DeeplabV3 [19] by approximately 4%, and the FCN [81] model by roughly 3%.

7.1.4 Augmentation Strategy For Sparse Datasets

In Chapter 6, we planned to answer research question 4 and reinforce the findings of the previous chapters regarding questions 2 and 3 (as outlined in section 1.1). To do this, we combined three methods to build a robust augmentation policy suitable for sparse datasets and applied them to semantic segmentation tasks. Our strategy is based on SPADE [92], a spatially-adaptive normalisation model that synthesises photorealistic images given a semantic input layout.

We demonstrated that our framework could improve results across three semantic segmentation tasks using GAN-derived synthetic images. We also seek to show how a low data density situation can be correlated with the improvement brought by GAN-based augmentation algorithms: When the dataset is sparse, it has gaps that the model can learn to fill. Consequently, augmented data is more effective in bringing performance improvements. Our method can effectively fill in these training data gaps by providing synthetic data that leads to improved performance,

resulting in a mean intersection-over-union (mIoU) score improvement of 0.5 to 14 percentage points.

7.2 Future Work

GANs have proven to be a powerful machine learning method for data augmentation and have achieved significant success in this task [14, 82, 72, 135]. In the quest to develop a strategy for obtaining a suitable augmentation method, this thesis covered some research activities like texture synthesis and arbitrary size image generation, all based on the GAN framework. This research can be further explored in future work to improve current performance. To do so, we believe we must answer the following questions.

To what extent does enhancing the quality of synthetic images hinder the augmentation capability of the model? To augment a dataset successfully, we know that the synthetic image must come from the same distribution as the dataset. However, can improving the image quality too much degrade the model augmentation capability?

We demonstrated that the data density could harm the model augmentation ability because, in a dense dataset, there are fewer gaps to fill. How far can we teach a model to overcome this restriction and learn to extend the dataset boundaries instead of just filling gaps in the existing data? As we know, a GAN model learns to imitate the dataset data distribution. Therefore is such a task possible?

In conclusion, future research works could focus on exploring novel techniques to optimise the performance and efficiency of GAN-based data augmentation methods, thereby advancing their applicability and impact across diverse domains.

Bibliography

[1] A.Dosovitskiy, J.T.Springenberg, and T.Brox. "Learning to Generate Chairs with Convolutional Neural Networks". In: *IEEE International Conference on Computer Vision and Pattern Recognition (CVPR)*. 2015.

[2] Antreas Antoniou, Amos Storkey, and Harrison Edwards. "Data augmentation generative adverdarial networks". In: *stat* 1050 (2018), p. 8.

[3] Antreas Antoniou, Amos Storkey, and Harrison Edwards. "Data augmentation generative adversarial networks". In: *arXiv preprint arXiv:1711.04340* (2017).

[4] Renato B Arantes, George Vogiatzis, and Diego R Faria. "CSC-GAN: Cycle and Semantic Consistency for Dataset Augmentation". In: *International Symposium on Visual Computing*. Springer. 2020, pp. 170–181.

[5] Renato B Arantes, George Vogiatzis, and Diego R Faria. "Learning an augmentation strategy for sparse datasets". In: *Image and Vision Computing* 117 (2022), p. 104338.

[6] Renato B Arantes, George Vogiatzis, and Diego R Faria. "rcGAN: Learning a Generative Model for Arbitrary Size Image Generation". In: *International Symposium on Visual Computing*. Springer. 2020, pp. 80–94.

[7] Renato Barros Arantes, George Vogiatzis, and Diego Faria. "QuiltGAN: An Adversarially Trained, Procedural Algorithm for Texture Generation". In: *International Conference on Computer Vision Systems*. Springer. 2019, pp. 423–432.

[8] Karim Armanious et al. "MedGAN: Medical image translation using GANs". In: *Computerized Medical Imaging and Graphics* 79 (2020), p. 101684.

[9] Dzmitry Bahdanau, Kyunghyun Cho, and Yoshua Bengio. "Neural machine translation by jointly learning to align and translate". In: *arXiv preprint arXiv:1409.0473* (2014).

[10] Lennart Bargsten and Alexander Schlaefer. "SpeckleGAN: a generative adversarial network with an adaptive speckle layer to augment limited training data for ultrasound image processing". In: *International journal of computer assisted radiology and surgery* 15.9 (2020), pp. 1427–1436.

[11] Connelly Barnes et al. "PatchMatch: A Randomized Correspondence Algorithm for Structural Image Editing". In: *ACM Transactions on Graphics (Proc. SIGGRAPH)* 28.3 (Aug. 2009).

[12] David Bau et al. "Semantic photo manipulation with a generative image prior". In: *arXiv preprint arXiv:2005.07727* (2020).

[13] Urs Bergmann, Nikolay Jetchev, and Roland Vollgraf. "Learning Texture Manifolds with the Periodic Spatial GAN". In: *Proceedings of the 34th International Conference on Machine Learning*. Ed. by Doina Precup and Yee Whye Teh. Vol. 70. Proceedings of Machine Learning Research. International Convention Centre, Sydney, Australia: PMLR, 2017, pp. 469–477.

[14] Jordan J Bird et al. "Fruit quality and defect image classification with conditional GAN data augmentation". In: *Scientia Horticulturae* 293 (2022), p. 110684.

[15] Christopher Bowles et al. "GAN Augmentation: Augmenting Training Data using Generative Adversarial Networks". In: *CoRR* abs/1810.10863 (2018). arXiv: 1810.10863.

[16] Andrew Brock, Jeff Donahue, and Karen Simonyan. "Large scale GAN training for high fidelity natural image synthesis". In: *arXiv preprint arXiv:1809.11096* (2018).

[17] Alexander Buslaev et al. "Albumentations: Fast and Flexible Image Augmentations". In: *Information* 11.2 (2020). ISSN: 2078-2489. DOI: 10.3390/info11020125. URL: https://www.mdpi.com/2078-2489/11/2/125.

[18] Liang-Chieh Chen et al. "Deeplab: Semantic image segmentation with deep convolutional nets, atrous convolution, and fully connected crfs". In: *IEEE transactions on pattern analysis and machine intelligence* 40.4 (2017), pp. 834–848.

[19] Liang-Chieh Chen et al. "Encoder-Decoder with Atrous Separable Convolution for Semantic Image Segmentation". In: *ECCV*. 2018.

[20] Liang-Chieh Chen et al. "Rethinking Atrous Convolution for Semantic Image Segmentation". In: (2017). arXiv: 1706.05587. URL: http://arxiv.org/abs/1706.05587.

[21] Tao Chen et al. "Sketch2Photo: Internet Image Montage". In: *ACM SIGGRAPH Asia 2009 Papers*. SIGGRAPH Asia '09. Yokohama, Japan: ACM, 2009, 124:1–124:10. ISBN: 978-1-60558-858-2.

[22] Jianpeng Cheng, Li Dong, and Mirella Lapata. "Long short-term memory-networks for machine reading". In: *arXiv preprint arXiv:1601.06733* (2016).

[23] Yunjey Choi et al. "StarGAN: Unified Generative Adversarial Networks for Multi-Domain Image-to-Image Translation". In: *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*. 2018.

[24] Yunjey Choi et al. "StarGAN v2: Diverse Image Synthesis for Multiple Domains". In: *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*. 2020.

[25] Dan Ciregan, Ueli Meier, and Jürgen Schmidhuber. "Multi-column deep neural networks for image classification". In: *2012 IEEE conference on computer vision and pattern recognition*. IEEE. 2012, pp. 3642–3649.

[26] Dan C Cireşan et al. "High-performance neural networks for visual object classification". In: *arXiv preprint arXiv:1102.0183* (2011).

[27] Marius Cordts et al. "The Cityscapes Dataset for Semantic Urban Scene Understanding". In: *Proc. of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*. 2016.

[28] Ekin D Cubuk et al. "Autoaugment: Learning augmentation strategies from data". In: *Proceedings of the IEEE conference on computer vision and pattern recognition*. 2019, pp. 113–123.

[29] Ekin D Cubuk et al. "Randaugment: Practical automated data augmentation with a reduced search space". In: *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition Workshops*. 2020, pp. 702–703.

[30] Jifeng Dai, Kaiming He, and Jian Sun. "Instance-aware semantic segmentation via multi-task network cascades". In: *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*. 2016, pp. 3150–3158.

[31] Ugur Demir and Gözde B. Ünal. "Patch-Based Image Inpainting with Generative Adversarial Networks". In: *CoRR* abs/1803.07422 (2018). arXiv: 1803.07422.

[32] Jeff Donahue and Karen Simonyan. "Large scale adversarial representation learning". In: *Advances in Neural Information Processing Systems*. 2019, pp. 10542–10552.

[33] DC Dowson and BV Landau. "The Fréchet distance between multivariate normal distributions". In: *Journal of multivariate analysis* 12.3 (1982), pp. 450–455.

[34] Alexei A. Efros and William T. Freeman. "Image Quilting for Texture Synthesis and Transfer". In: *Proceedings of the 28th Annual Conference on Computer Graphics and Interactive Techniques*. SIGGRAPH '01. New York, NY, USA: ACM, 2001, pp. 341–346. ISBN: 1-58113-374-X.

[35] Alexei A. Efros and Thomas K. Leung. "Texture Synthesis by Non-Parametric Sampling". In: *Proceedings of the International Conference on Computer Vision-Volume 2 - Volume 2*. ICCV '99. Washington, DC, USA: IEEE Computer Society, 1999, pp. 1033–. ISBN: 0-7695-0164-8.

[36] Hao-Shu Fang et al. "Instaboost: Boosting instance segmentation via probability map guided copy-pasting". In: *Proceedings of the IEEE International Conference on Computer Vision*. 2019, pp. 682–691.

[37] Maurice Fréchet. "Sur la distance de deux lois de probabilité". In: *Comptes Rendus Hebdomadaires des Seances de L Academie des Sciences* 244.6 (1957), pp. 689–692.

[38] Anna Frühstück, Ibraheem Alhashim, and Peter Wonka. "Tilegan: synthesis of large-scale non-homogeneous textures". In: *ACM Transactions on Graphics (TOG)* 38.4 (2019), pp. 1–11.

[39] Leon A. Gatys, Alexander S. Ecker, and Matthias Bethge. "Texture Synthesis Using Convolutional Neural Networks". In: *Proceedings of the 28th International Conference on Neural Information Processing Systems - Volume 1*. NIPS'15. Montreal, Canada: MIT Press, 2015, pp. 262–270.

[40] Golnaz Ghiasi et al. "Exploring the structure of a real-time, arbitrary neural artistic stylization network". In: *arXiv preprint arXiv:1705.06830* (2017).

[41] Raphael Gontijo-Lopes et al. "Tradeoffs in Data Augmentation: An Empirical Study". In: *International Conference on Learning Representations*. 2020.

[42] Ian Goodfellow. "NIPS 2016 Tutorial: Generative Adversarial Networks". In: (2016). ISSN: 978-3-319-10589-5.

[43] Ian Goodfellow et al. "Generative Adversarial Nets". In: *Advances in Neural Information Processing Systems 27*. Ed. by Z. Ghahramani et al. Curran Associates, Inc., 2014, pp. 2672–2680.

[44] Dongyoon Han, Jiwhan Kim, and Junmo Kim. "Deep pyramidal residual networks". In: *Proceedings of the IEEE conference on computer vision and pattern recognition*. 2017, pp. 5927–5935.

[45] Bharath Hariharan et al. "Semantic contours from inverse detectors". In: *2011 International Conference on Computer Vision*. IEEE. 2011, pp. 991–998.

[46] James Hays and Alexei A. Efros. "Scene Completion Using Millions of Photographs". In: *ACM SIGGRAPH 2007 Papers*. SIGGRAPH '07. San Diego, California: ACM, 2007.

[47] Kaiming He et al. "Deep Residual Learning for Image Recognition". In: *2016 IEEE Conference on Computer Vision and Pattern Recognition, CVPR 2016, Las Vegas, NV, USA, June 27-30, 2016*. 2016, pp. 770–778.

[48] Kaiming He et al. "Mask r-cnn". In: *Proceedings of the IEEE international conference on computer vision*. 2017, pp. 2961–2969.

[49] Martin Heusel et al. "Gans trained by a two time-scale update rule converge to a local nash equilibrium". In: *Advances in neural information processing systems* 30 (2017).

[50] Tobias Hinz et al. "Improved Techniques for Training Single-Image GANs". In: *Proceedings of the IEEE/CVF Winter Conference on Applications of Computer Vision (WACV)*. 2021, pp. 1300–1309.

[51] Judy Hoffman et al. "Cycada: Cycle-consistent adversarial domain adaptation". In: *International conference on machine learning*. 2018, pp. 1989–1998.

[52] Jie Hu, Li Shen, and Gang Sun. "Squeeze-and-excitation networks". In: *Proceedings of the IEEE conference on computer vision and pattern recognition*. 2018, pp. 7132–7141.

[53] Zilong Huang et al. "CCNet: Criss-Cross Attention for Semantic Segmentation". In: *IEEE Transactions on Pattern Analysis and Machine Intelligence* (2020), pp. 1–1. DOI: 10.1109/TPAMI.2020.3007032.

[54] Satoshi Iizuka, Edgar Simo-Serra, and Hiroshi Ishikawa. "Globally and Locally Consistent Image Completion". In: *ACM Trans. Graph.* 36.4 (July 2017), 107:1–107:14. ISSN: 0730-0301.

[55] Phillip Isola et al. "Image-to-Image Translation with Conditional Adversarial Networks". In: *2017 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)* (2017), pp. 5967–5976.

[56] Phillip Isola et al. "Image-to-image translation with conditional adversarial networks". In: *Proceedings of the IEEE conference on computer vision and pattern recognition*. 2017, pp. 1125–1134.

[57] Philip T Jackson et al. "Style Augmentation: Data Augmentation via Style Randomization". In: *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition Workshops*. 2019, pp. 83–92.

[58] Nikolay Jetchev, Urs Bergmann, and Roland Vollgraf. "Texture Synthesis with Spatial Generative Adversarial Networks". In: *CoRR* abs/1611.08207 (2016). arXiv: 1611.08207.

[59] Hongsheng Jin et al. "A deep 3D residual CNN for false-positive reduction in pulmonary nodule detection". In: *Medical physics* 45.5 (2018), pp. 2097–2107.

[60] Tero Karras, Samuli Laine, and Timo Aila. "A Style-Based Generator Architecture for Generative Adversarial Networks." In: abs/1812.04948 (2018).

[61] Tero Karras, Samuli Laine, and Timo Aila. "A style-based generator architecture for generative adversarial networks". In: *Proceedings of the IEEE conference on computer vision and pattern recognition*. 2019, pp. 4401–4410.

[62] Tero Karras et al. "Progressive growing of gans for improved quality, stability, and variation". In: *arXiv preprint arXiv:1710.10196* (2017).

[63] Tero Karras et al. "Progressive Growing of GANs for Improved Quality, Stability, and Variation". In: *International Conference on Learning Representations*. 2018.

[64] Diederik P Kingma and Jimmy Ba. "Adam: A method for stochastic optimization". In: *arXiv preprint arXiv:1412.6980* (2014).

[65] Diederik P. Kingma and Max Welling. "Auto-Encoding Variational Bayes". In: *2nd International Conference on Learning Representations, ICLR 2014, Banff, AB, Canada, April 14-16, 2014, Conference Track Proceedings*. 2014.

[66] Alex Krizhevsky, Ilya Sutskever, and Geoffrey E Hinton. "Imagenet classification with deep convolutional neural networks". In: *Advances in neural information processing systems*. 2012, pp. 1097–1105.

[67] Alex Krizhevsky, Ilya Sutskever, and Geoffrey E Hinton. "Imagenet classification with deep convolutional neural networks". In: *Communications of the ACM* 60.6 (2017), pp. 84–90.

[68] Ashnil Kumar et al. "Chapter Five - Machine learning in medical imaging". In: *Biomedical Information Technology (Second Edition)*. Ed. by David Dagan Feng. Second Edition. Biomedical Engineering. Academic Press, 2020, pp. 167–196. ISBN: 978-0-12-816034-3. DOI: https://doi.org/10.1016/B978-0-12-816034-3.00005-5. URL: https://www.sciencedirect.com/science/article/pii/B9780128160343000055.

[69] Yann LeCun et al. "Gradient-based learning applied to document recognition". In: *Proceedings of the IEEE* 86.11 (1998), pp. 2278–2324.

[70] C. Ledig et al. "Photo-Realistic Single Image Super-Resolution Using a Generative Adversarial Network". In: *2017 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*. 2017, pp. 105–114.

[71] Joseph Lemley, Shabab Bazrafkan, and Peter Corcoran. "Smart augmentation learning an optimal data augmentation strategy". In: *Ieee Access* 5 (2017), pp. 5858–5869.

[72] Menglu Li and Wen Zhang. "PHIAF: prediction of phage-host interactions with GAN-based data augmentation and sequence-based feature fusion". In: *Briefings in Bioinformatics* 23.1 (2022), bbab348.

[73] Yi Li et al. "Fully convolutional instance-aware semantic segmentation". In: *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*. 2017, pp. 2359–2367.

[74] Sungbin Lim et al. "Fast autoaugment". In: *Advances in Neural Information Processing Systems*. 2019, pp. 6665–6675.

[75] Chen Lin et al. "Online Hyper-Parameter Learning for Auto-Augmentation Strategy". In: *Proceedings of the IEEE/CVF International Conference on Computer Vision (ICCV)*. 2019.

[76] Chen Lin et al. "Online hyper-parameter learning for auto-augmentation strategy". In: *Proceedings of the IEEE International Conference on Computer Vision*. 2019, pp. 6579–6588.

[77] Min Lin, Qiang Chen, and Shuicheng Yan. "Network in network". In: *arXiv preprint arXiv:1312.4400* (2013).

[78] Tsung-Yi Lin et al. "Feature pyramid networks for object detection". In: *Proceedings of the IEEE conference on computer vision and pattern recognition*. 2017, pp. 2117–2125.

[79] Ming-Yu Liu et al. "Few-Shot Unsupervised Image-to-Image Translation". In: *Proceedings of the IEEE/CVF International Conference on Computer Vision (ICCV)*. 2019.

[80] Wei Liu et al. "Ssd: Single shot multibox detector". In: *European conference on computer vision*. Springer. 2016, pp. 21–37.

[81] Jonathan Long, Evan Shelhamer, and Trevor Darrell. "Fully convolutional networks for semantic segmentation". In: *Proceedings of the IEEE conference on computer vision and pattern recognition*. 2015, pp. 3431–3440.

[82] Fei Ma et al. "Data Augmentation for Audio-Visual Emotion Recognition with an Efficient Multimodal Conditional GAN". In: *Applied Sciences* 12.1 (2022), p. 527.

[83] Laurens van der Maaten and Geoffrey Hinton. "Visualizing data using t-SNE". In: *Journal of machine learning research* 9.Nov (2008), pp. 2579–2605.

[84] Xudong Mao et al. "Least Squares Generative Adversarial Networks". In: *IEEE International Conference on Computer Vision, ICCV 2017, Venice, Italy, October 22-29, 2017*. 2017, pp. 2813–2821.

[85] Giovanni Mariani et al. "BAGAN: Data Augmentation with Balancing GAN". In: *CoRR* abs/1803.09655 (2018). arXiv: 1803.09655.

[86] Kazuhisa Matsunaga et al. "Image classification of melanoma, nevus and seborrheic keratosis by deep neural network ensemble". In: *arXiv preprint arXiv:1703.03108* (2017).

[87] Mehdi Mirza and Simon Osindero. "Conditional Generative Adversarial Nets". In: *CoRR* abs/1411.1784 (2014). arXiv: 1411.1784.

[88] Michael Oechsle et al. "Texture fields: Learning texture representations in function space". In: *Proceedings of the IEEE/CVF International Conference on Computer Vision*. 2019, pp. 4531–4540.

[89] Aaron van den Oord, Yazhe Li, and Oriol Vinyals. "Representation learning with contrastive predictive coding". In: *arXiv preprint arXiv:1807.03748* (2018).

[90] Taesung Park et al. "Contrastive learning for unpaired image-to-image translation". In: *European Conference on Computer Vision*. Springer. 2020, pp. 319–345.

[91] Taesung Park et al. "Contrastive Learning for Unpaired Image-to-Image Translation". In: *European Conference on Computer Vision*. 2020.

[92] Taesung Park et al. "Semantic Image Synthesis with Spatially-Adaptive Normalization". In: *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*. 2019.

[93] Magdalini Paschali et al. "Data augmentation with manifold exploring geometric transformations for increased performance and robustness". In: *arXiv preprint arXiv:1901.04420* (2019).

[94] Deepak Pathak et al. "Context Encoders: Feature Learning by Inpainting". In: 2016.

[95] Kexin Pei et al. "Deepxplore: Automated whitebox testing of deep learning systems". In: *proceedings of the 26th Symposium on Operating Systems Principles*. 2017, pp. 1–18.

[96] Kexin Pei et al. "Towards practical verification of machine learning: The case of computer vision systems". In: *arXiv preprint arXiv:1712.01785* (2017).

[97] Tiziano Portenier, Siavash Bigdeli, and Orcun Goksel. "Gramgan: Deep 3d texture synthesis from 2d exemplars". In: *arXiv preprint arXiv:2006.16112* (2020).

[98] Javier Portilla and Eero P. Simoncelli. "A Parametric Texture Model Based on Joint Statistics of Complex Wavelet Coefficients". In: *International Journal of Computer Vision* 40.1 (2000), pp. 49–70. ISSN: 09205691.

[99] Jerry L. Prince et al. "Chapter 1 - Image synthesis and superresolution in medical imaging". In: *Handbook of Medical Image Computing and Computer Assisted Intervention*. Ed. by S. Kevin Zhou, Daniel Rueckert, and Gabor Fichtinger. The Elsevier and MICCAI Society Book Series. Academic Press, 2020, pp. 1–24. ISBN: 978-0-12-816176-0. DOI: https://doi.org/10.1016/B978-0-12-816176-0.00006-5. URL: https://www.sciencedirect.com/science/article/pii/B9780128161760000065.

[100] Charles R Qi et al. "Pointnet: Deep learning on point sets for 3d classification and segmentation". In: *Proceedings of the IEEE conference on computer vision and pattern recognition*. 2017, pp. 652–660.

[101] Alec Radford, Luke Metz, and Soumith Chintala. "Unsupervised Representation Learning with Deep Convolutional Generative Adversarial Networks". In: abs/1511.06434 (2016).

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

[103] Radim Šára Radim Tyleček. "Spatial Pattern Templates for Recognition of Objects with Regular Structure". In: *Proc. GCPR*. Saarbrucken, Germany, 2013.

[104] Esteban Real et al. "Regularized evolution for image classifier architecture search". In: *Proceedings of the aaai conference on artificial intelligence*. Vol. 33. 2019, pp. 4780–4789.

[105] Joseph Redmon et al. "You only look once: Unified, real-time object detection". In: *Proceedings of the IEEE conference on computer vision and pattern recognition*. 2016, pp. 779–788.

[106] Shaoqing Ren et al. "Faster r-cnn: Towards real-time object detection with region proposal networks". In: *Advances in neural information processing systems*. 2015, pp. 91–99.

[107] Tamar Rott Shaham, Tali Dekel, and Tomer Michaeli. "SinGAN: Learning a Generative Model from a Single Natural Image". In: *Computer Vision (ICCV), IEEE International Conference on*. 2019.

[108] Veit Sandfort et al. "Data augmentation using generative adversarial networks (CycleGAN) to improve generalizability in CT segmentation tasks". In: *Scientific reports* 9.1 (2019), pp. 1–9.

[109] Divya Shanmugam et al. "When and why test-time augmentation works". In: *arXiv preprint arXiv:2011.11156* (2020).

[110] Evan Shelhamer, Jonathan Long, and Trevor Darrell. "Fully Convolutional Networks for Semantic Segmentation". In: *IEEE Trans. Pattern Anal. Mach. Intell.* 39.4 (Apr. 2017), pp. 640–651. ISSN: 0162-8828.

[111] Evan Shelhamer, Jonathan Long, and Trevor Darrell. "Fully Convolutional Networks for Semantic Segmentation". In: *IEEE Trans. Pattern Anal. Mach. Intell.* 39.4 (Apr. 2017), pp. 640–651. ISSN: 0162-8828.

[112] Assaf Shocher et al. “InGAN: Capturing and Retargeting the "DNA" of a Natural Image”. In: *The IEEE International Conference on Computer Vision (ICCV)*. 2019.

[113] Connor Shorten and Taghi M Khoshgoftaar. “A survey on image data augmentation for deep learning”. In: *Journal of Big Data* 6.1 (2019), p. 60.

[114] Ashish Shrivastava et al. “Learning from Simulated and Unsupervised Images through Adversarial Training”. In: *CoRR* abs/1612.07828 (2016). arXiv: 1612.07828.

[115] Patrice Y Simard, David Steinkraus, John C Platt, et al. “Best practices for convolutional neural networks applied to visual document analysis.” In: *Icdar*. Vol. 3. 2003. 2003.

[116] Karen Simonyan and Andrew Zisserman. “Very deep convolutional networks for large-scale image recognition”. In: *arXiv preprint arXiv:1409.1556* (2014).

[117] Leon Sixt, Benjamin Wild, and Tim Landgraf. “Rendergan: Generating realistic labeled data”. In: *Frontiers in Robotics and AI* 5 (2018), p. 66.

[118] Shuran Song, Samuel P Lichtenberg, and Jianxiong Xiao. “Sun rgb-d: A rgb-d scene understanding benchmark suite”. In: *Proceedings of the IEEE conference on computer vision and pattern recognition*. 2015, pp. 567–576.

[119] Vadim Sushko, Jurgen Gall, and Anna Khoreva. “One-shot gan: Learning to generate samples from single images and videos”. In: *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*. 2021, pp. 2596–2600.

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

[121] 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.

[122] Nima Tajbakhsh et al. “Embracing imperfect datasets: A review of deep learning solutions for medical image segmentation”. In: *Medical Image Analysis* 63 (2020), p. 101693.

[123] Mingxing Tan and Quoc Le. "Efficientnet: Rethinking model scaling for convolutional neural networks". In: *International Conference on Machine Learning*. PMLR. 2019, pp. 6105–6114.

[124] Mingxing Tan, Ruoming Pang, and Quoc V Le. "Efficientdet: Scalable and efficient object detection". In: *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*. 2020, pp. 10781–10790.

[125] Toan Tran et al. "A bayesian data augmentation approach for learning deep models". In: *Advances in neural information processing systems*. 2017, pp. 2797–2806.

[126] Leonid Nisonovich Vaserstein. "Markov processes over denumerable products of spaces, describing large systems of automata". In: *Problemy Peredachi Informatsii* 5.3 (1969), pp. 64–72.

[127] Yael Vinker et al. *Deep Single Image Manipulation*. 2020. arXiv: 2007.01289.

[128] Li Wan et al. "Regularization of neural networks using dropconnect". In: *International conference on machine learning*. 2013, pp. 1058–1066.

[129] Ting-Chun Wang et al. "High-resolution image synthesis and semantic manipulation with conditional gans". In: *Proceedings of the IEEE conference on computer vision and pattern recognition*. 2018, pp. 8798–8807.

[130] Xiaolong Wang and Abhinav Gupta. "Generative image modeling using style and structure adversarial networks". In: *European conference on computer vision*. Springer. 2016, pp. 318–335.

[131] Wenqi Xian et al. "Texturegan: Controlling deep image synthesis with texture patches". In: *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*. 2018, pp. 8456–8465.

[132] Wenqiang Xu et al. "Explicit shape encoding for real-time instance segmentation". In: *Proceedings of the IEEE International Conference on Computer Vision*. 2019, pp. 5168–5177.

[133] Hui Ye et al. "Improving Text-to-Image Synthesis Using Contrastive Learning". In: *arXiv preprint arXiv:2107.02423* (2021).

[134] Xin Yi, Ekta Walia, and Paul Babyn. "Generative adversarial network in medical imaging: A review". In: *Medical Image Analysis* 58 (2019), p. 101552. ISSN: 1361-8415.

[135] Aiming Zhang et al. "EEG data augmentation for emotion recognition with a multiple generator conditional Wasserstein GAN". In: *Complex & Intelligent Systems* 8.4 (2022), pp. 3059–3071.

[136] Han Zhang et al. "Self-attention generative adversarial networks". In: *International Conference on Machine Learning*. 2019, pp. 7354–7363.

[137] Han Zhang et al. "Stackgan++: Realistic image synthesis with stacked generative adversarial networks". In: *IEEE transactions on pattern analysis and machine intelligence* 41.8 (2018), pp. 1947–1962.

[138] Xin Zhao et al. "Solid texture synthesis using generative adversarial networks". In: *arXiv preprint arXiv:2102.03973* (2021).

[139] Jun-Yan Zhu et al. "Generative Visual Manipulation on the Natural Image Manifold". In: *Proceedings of European Conference on Computer Vision (ECCV)*. 2016.

[140] Jun-Yan Zhu et al. "Unpaired Image-to-Image Translation using Cycle-Consistent Adversarial Networks". In: *Computer Vision (ICCV), 2017 IEEE International Conference on*. 2017.

[141] Xinyue Zhu et al. "Emotion classification with data augmentation using generative adversarial networks". In: *Pacific-Asia conference on knowledge discovery and data mining*. Springer. 2018, pp. 349–360.

[142] Barret Zoph et al. "Learning transferable architectures for scalable image recognition". In: *Proceedings of the IEEE conference on computer vision and pattern recognition*. 2018, pp. 8697–8710.