# Realization of a diagnostic aid in orthodontics by deep learning

Tien Tai Doan

université
PARIS-SACLAY

# Réalisation d'une Aide au Diagnostic en Orthodontie par Apprentissage Profond

## *Realization of a Diagnostic Aid in Orthodontics by Deep Learning*

**Thèse de doctorat de l'Université Paris-Saclay**

École doctorale numéro 580, Sciences et Technologies de l'Information et de la Communication, STIC
Spécialité de doctorat: Informatique
Unité de recherche: IBISC EA 4526 Informatique, Biologie Intégrative Systèmes Complexes
Référent: Université d'Évry Val d'Essonne

**Thèse présentée et soutenue à Évry-Courcouronnes le 2 juillet 2021, par**

**Tien Tai DOAN**

**Composition du jury:**

| | |
|---|---|
| **Francois Xavier JOLLOIS**<br>Maître de Conférences, Université de Paris | President |
| **Nataliya SOKOLOVSKA**<br>Maitresse de Conférence, Sorbonne Université | Rapporteur et examinateur |
| **Younes BENNANI**<br>Professeur, Université Sorbonne Paris Nord | Rapporteur et examinateur |
| **Hedi TABIA**<br>Professeur, Université Paris-Saclay | Examinateur |
| **Blaise HANCZAR**<br>Professeur, Université Paris-Saclay | Directeur |
| **Guillaume GHYSELINCK**<br>R&D Director, Dental Mind company | Co-encadrant |

Thèse de doctorat

NNT: 2021UPASG033

# Acknowledgements

First of all, I would like to thank my supervisors Guillaume Ghyselinck and Prof. Hanczar Blaise, for their countless support, guidance and encouragement during three years of preparing for my thesis. I remember after our first interview in June 2017 at Montreuil, where we had an interesting discussion on the thesis description and generative adversarial networks, it took us about nine months to get all the paperwork done. However, after three years of working with them, I am sure that it was worth waiting.

Many thanks to members of the thesis jury Prof. Nataliya Sokolovska, Prof. Younes Bennani, Prof. Francois Xavier Jollois, and Prof. Hedi Tabia for spending their time reading my thesis manuscript, listening to my presentation and giving intuitive feedback.

I thank the Dental Mind company, especially the R&D department. Particularly, thanks to Laurent Debraux for his advice and inspirations. I managed to rent my first apartment in Paris thanks to his guarantee. I also remember when Guillaume spent his morning at the Paris prefecture to help me with the paperwork. I thank my colleague and good friend, Trung, who welcomed me to his place when I arrived in France. Many thanks to Robert, Chenyu, Kawtar, Nicolas, Macro, and other team members for their enthusiasm, intelligence, help, and friendliness. Thanks to Clement and Eliott for their super-speed technical support.

I also would like to thank laboratory IBISC, Paris-Saclay University, and my colleagues there. We have not had much time together, but I always feel welcomed when I get there. Thanks to Murielle for always being helpful in so many ways.

Special thanks to Prof. Marie Luong and Prof. Yann Chevaleyre for their guidance and support during and after my master's internship and their effort to help me find a thesis.

I am thankful for co Dao, Anthony, anh Le, and Long for supporting me so much to adapt to life in France when I first came to this wonderful country.

With lots of joy, I would like to express my gratitude to my housemates in a Christian house at Suresnes, whom I call brothers and sisters, who make me realize the value and the power of love in this life. I am so touched by all the love, care and patience they had for me, especially when I got sick because of the Covid-19. Big thanks to Christiane and Bertrand for starting this "perfectly-imperfect" house more than thirty years ago. I especially thank Gwenael, Dimitri and Nicolas for insightful conversations, for valuable advice, for being examples of caring and serving others, and of course, for their endless sense of humor. Thank you, beloved Alexandra. You remind me to let go of the burdens of the past, worry less about the future, and enjoy the little joys in the present.

Warmly thanks my brothers and sisters at the Nhan Cap church in Hanoi, who grew up with me and gave me so much support, financially and spiritually. I am so grateful for their love, prayers and encouragement. They always make up a big part of my heart.

From the bottom of my heart, thanks to my parents Joseph and Xuan, and my brother Josh. Thank you for loving me as I am, for never giving up on me, and for being there when I needed you most. Millions of thanks to mom and dad for sacrificing so much for me. Josh, I am proud of you, and I am happy to see the intelligent and courageous man you have become. I also thank my grandparents, aunts, uncles, and cousins for their supports and encouragement.

Finally, I give all the glory, greatness, and praise to my Lord Jesus Christ, for His love, grace, and blessings in my life, especially during my last three years preparing for this thesis.

# Contents

# List of Figures

# List of Tables

# Chapter 1

# Introduction

Orthodontics is one of the dentistry fields that focuses on the development of teeth, dental occlusion, and facial growth. It is concerned with the diagnosis, prevention, interception, and treatments of malocclusion. Orthodontics mostly works on the craniofacial skeleton, especially teeth and the alveolar bone supporting the teeth.

One important key to orthodontic treatment's success is the accuracy of the diagnosis and the treatment plan. A small mistake in either the diagnosis or the planning may lead to dramatic cost, not only in terms of money and time, but it may cause serious consequences to the patient's health. Therefore, orthodontists usually have to consider many factors such as the patient's medical history, dental history, examinations, radiographs, and even 3d-models to clearly understand the patient's condition before starting orthodontic treatment. Among these factors, radiographs seem to contribute the most to the orthodontic diagnosis because they reveal the structure of tooth roots, craniofacial skeleton, and nerves hidden from human eyes.

Along with the development of science and technology in general, the dental industry has also made significant achievements and progress in applying modern technologies to the diagnostic process and treatment regimen. The application of artificial intelligence (AI), especially deep learning (DL), or convolutional neural networks (CNNs), has become more popular in the medical field than ever before. In the field of dentistry, deep learning models also have their advantages by working with images. They have been used for recognizing, classifying, and segmenting different components in multiple types of photographs, especially radiographic images [114] [24] [163]. Furthermore, deep learning algorithms also contribute to the diagnosis process by detecting and recognizing dental complications [98] [91] [4].

Despite the promising performance of deep learning in processing dental images, as reported in these studies, the application of CNNs as a diagnostic aid in orthodontics is facing multiple obstacles. Firstly, the type of data is limited to radiographic images. There is no machine learning model trained to recognize dental elements or detect gum diseases in images captured by common devices such as smartphone-cameras, to the limit of our knowledge at the moment of writing. Besides, data like radiographs can only be captured by expensive professional devices, so it is difficult to collect many samples for training convolutional networks. This explains why the datasets used in these papers are very limited. Moreover, current applications of deep learning in dentistry and orthodontics are stopping at solving one single task like classification, recognition, segmentation, etc. In contrast, it takes a series of networks and algorithms to make a product or service that solve a real-life problem. Therefore, studying and realizing deep learning algorithms for orthodontic diagnosis with a large dataset that can be easily collected than radiographs would be a worth-considering contribution.

## 1.1 Objectives & Contributions

This thesis aims to study the latest technologies of deep learning and apply them to aid in orthodontic imaging. Instead of processing all types of images used in the orthodontic field, we focus on color images taken by smartphones through the Dental Monitoring application of the company Dental Mind. This is the first company in the world to monitor orthodontic progress and detailed oral health assessment via smartphone. Orthodontics is a complex process involving many steps before, during and after the treatment. Within the limits of this thesis, deep learning technology is applied to serve the image processing and diagnosis of common diseases in the orthodontic process. We also study the problems neural networks encounter while working with dental imaging and propose remedies. The contributions of the thesis are as follows:

### 1.1.1 Diagnosis Using Photos Captured by Mobile Phones

Several studies have shown the potential of using deep neural networks to detect problems in teeth and gums. Juan et al.'s work [82] applies computer vision methods to diagnose periodontal diseases by measuring depth probing automatically. A special camera obtained images fitted together with a dental probe. In [77], the authors used multi-view CNNs for detecting dental diseases from red fluorescent images. Rana et al. [129] used a convolutional autoencoder for segmenting inflammation from intraoral fluorescence images. Deep learning was also used on periapical radiographs to diagnose and predict periodontally compromised teeth in [99] and detection of dental caries in [98]. The common drawback of these methods is that they require x-rays or fluorescent techniques for obtaining images, which means patients still have to be diagnosed at clinics. To ease this process, we apply deep learning models directly to images captured by smartphone to detect problems such as gingivitis and crowding of teeth. This study is an important piece of the puzzle to make remote examination simpler and more accessible, especially for patients who live far away from the clinic, have to travel frequently, or are too busy to have appointments. Dentists, in particular orthodontists, also benefit from this approach as it saves them time, so they can spend more time with difficult cases, as well as serve more patients in a period of time. Despite the advantages, automatic diagnostics on images captured with a smartphone also encounter certain difficulties. Patients use their own smartphones to take pictures, so our database's images vary in size, lighting conditions, shooting angle, sharpness, etc. For this reason, before image diagnosis, we also propose a corresponding preprocessing, which includes image processing algorithms combined with pre-trained machine learning models. The preprocessing helps to increase the diagnosis accuracy. Different factors that may help increase the training performance, such as the use of multiple views and data augmentation techniques, are also studied in the chapter. Furthermore, we study the prediction features of the classifier using saliency maps.

### 1.1.2 CNN-based Domain Adaptation for Data Augmentation

Lack of training data is a pervasive problem when someone wants to apply deep learning to problem-solving in the medical field [115]. Dentistry and orthodontics are no exception. Schwendicke et al. [143] show that despite the great potential of machine learning in dentistry, constructing large datasets for training neural networks faces multiple barriers. It is difficult to find databases of dental images with the size of thousands of samples. In the scope of this thesis, we are allowed to study on the Dental Mind database. Having many images is not enough; they need to be carefully labeled to be used as training data, which takes time and effort. Therefore, we study methods for data augmentation. Most research on data augmentation in the medical field focuses on parameter selection and ordering image processing operations such as rotate, scale, crop, add noises, etc [74]. Recently, with the rapid development of adversarial generative networks (GANs) [56], many studies have been carried out to apply these models to medical

data augmentation to train deep learning models [46] [45]. Most of them tend to generate new samples from existing samples in the same image domain. One potential but not popular research direction to apply GANs-based domain adaptation model for augmenting training data. Our thesis studies different state-of-the-art image translation models and applies them to generate additional training samples to increase learning models' accuracy that diagnose dental problems. Specifically, we prepare a dataset containing 3d models of teeth, reconstructed from real images of teeth. This dataset gives us thousands of pairs of samples to train a GANs-based supervised image-to-image translation network that transforms projections of 3d teeth models to real-looking images. These synthesised images serve as additional training data. We also study multiple scenarios and factors that affect the domain adaptation model's performance, contributing to the outcome of the final diagnosis model.

### 1.1.3 Disentangled Unsupervised Image-to-Image Translation

Among domain adaptation methods, unsupervised image-to-image translation is widely used because it does not require paired samples, which is usually not accessible in this kind of problem. Thanks to this simplicity, the approach is applied for various image translation tasks, and different architectures have also been proposed. These extensions are mostly additional training constraints, such as the reconstruction of the original image [188], pixel-wise loss [148], semantic features [160], etc., which help to enhance the quality of output images. An often overlooked problem with unsupervised image translation models is the change in the structure of the image caused by the transformation, also called label-shift [101]. Because training images are not labeled, there is no constraint that stops label-shift from happening. This problem becomes serious when we apply one of these models for generating images in the medical domain. In [29], a common image translation model reportedly modifies details of tumors in images while translating them between two types of MR images. Having the ability to control features of the generated images is one way to address this problem. In [73], an unsupervised image-to-image translation method is introduced with a style vector that contains domain-irrelevant features of the output images. We study the features stored in this style vector and propose modifications in the method's architecture to learn meaningful features, so that we can control those features by adjusting the vector's value.

### 1.1.4 Generation of Aligners in Portraits

Besides learning to diagnose orthodontic images, we also have a chance to contribute to another work related to orthodontic treatments. We use deep learning technology to modify images to help patients see relatively how their teeth will look when they wear orthodontic appliances. Besides price, comfort and time, appearance is an important factor for a patient to decide on his orthodontic treatment when there are multiple possible options. Having a predicted image of their appearance during treatment will help the patient make a more suitable choice, minimizing the feeling of disappointment and the willingness to give up while wearing orthodontic appliances. In the scope of the thesis, we focus on only one kind of appliance: aligner, which has been widely used thanks to its convenience. Our output image criterion is sharpness, realism and high-resolution.

## 1.2 Thesis Outline

In Chapter 2, we explain certain basic dentistry concepts, the structure of teeth, common dental problems, especially diseases related to teeth and gums that often appear in orthodontic treatment. We then describe orthodontic treatment procedures with appliances involved. The chapter also covers orthodontic appliances-related problems that a remote monitoring system should be aware of. At the end of the chapter, we give a glimpse of some of Dental Mind's

services and products related to this thesis and discuss their problems. Note that the Dental Mind products and services descriptions in this thesis are not exhaustive representations but are only freakishly descriptive to help readers understand our work context.

Chapter 3 covers the state-of-the-art deep learning algorithms and technologies used in this thesis, such as object recognition and segmentation, normalization techniques, generative adversarial networks (GANs) and their evaluation metrics, image-to-image translation models. We also explain the background of some topics that are related to our proposed method: disentangled representation learning and few-shot domain adaption for classification. The end of the chapter mentions state-of-the-art achievements of deep learning in the domain of dentistry and orthodontics.

In Chapter 4, we train a deep neural network to detect the presence of gingivitis in oral images taken by smartphone cameras. To detect gingivitis at the level of teeth, we combine networks pre-trained at the host company to form a preprocessing pipeline. Our experiments show that gingivitis detection is improved when it is done on multiple views, while augmenting the training data using an unsupervised image translation technique does not improve the classification performance. From this project, we also learn that our neural network detects gingivitis based more on the gum's shape than its texture.

In Chapter 5, we generate training data by transforming the style of 3d-models of teeth to make them look realistic and then to use them as additional training data for detecting crowded teeth. We apply a state-of-the-art supervised image-to-image translation method to transform each view of a 3d teeth model into a colored image. Also, data augmentation is also implemented in the 3d environment by rotating the teeth models on the x, y and z axes. Experiments show that using synthetic data significantly improves learning performance but using too much synthetic data can be problematic. It also points out that 3d augmentation can slightly increase the classification accuracy.

In Chapter 6, we propose a new multimodal image translation method - InfoMUNIT - an extension of the state-of-the-art method MUNIT. Our method allows controlling the style of the generated images and improves their quality and diversity. It learns to maximize the mutual information between a subset of style code and the output images' distribution. Experiments show that our model can translate one image from the source domain to multiple images in the target domain and explore and manipulate features of the outputs without annotation. Furthermore, it achieves superior diversity and competitive image quality to state-of-the-art methods in multiple image translation tasks. We also propose F-InfoMUNIT, an extension of InfoMUNIT, for a few-shot domain adaptation and apply it for augmenting training data. The method produces a comparable performance compared to state-of-the-art methods in the field.

In Chapter 6, we generate plastic aligners on a high-resolution portrait of patients using two unsupervised image translation methods with two different architectures. The networks take images of teeth having no appliances as inputs and make them look like wearing transparent aligners. We compare the two methods in terms of generated image quality, resolution and memory consumption. Our best model contributes to a commercial product of Dental Mind, namely Vision (Section 2.4.3).

Chapter 8 is a summary of the thesis results. We also discuss the advantages and limitations of each work and suggest some directions for future work.

# Chapter 2

# Context

## 2.1 Dental problems

Dental health is important and greatly affects the health of the entire human body and its quality. Studies have shown a strong link between poor dental health and other diseases such as cardiovascular disease [36], pulmonary disease [112] and diabetes mellitus [93]. Improving dental health can also improve life quality. According to [121], patients having symptoms such as bad breath, toothache, sore gums, and so on suffer a lot in their lives physically and mentally. In this section, we describe common dental problems and discuss prevention and treatment.

### 2.1.1 Tooth decay

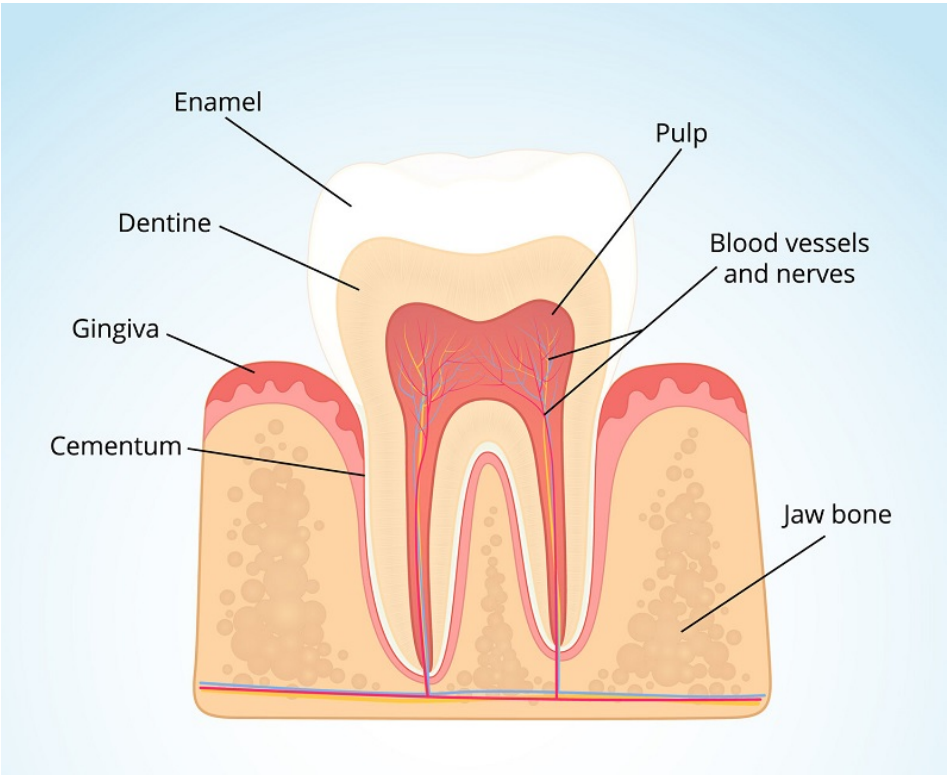



Figure 2.1: Structure of a tooth, image from [162]

Teeth are an important and complicated part of the human body. Each tooth consists of multiple components with different roles and characteristics. Enamel is the outermost layer of the tooth that covers the inner layers. It is made of calcium phosphate and other minerals making it white and strong. The enamel layer creates a sturdy sheath that protects the softer components and plays an essential role in crushing food before it reaches the esophagus and

stomach. Enamel cannot be regenerated once damaged, but it can be slowly built up thanks to minerals from saliva and fluoride from toothpaste or other sources. Dentin (or dentine) is the second hard layer of the tooth. Made up of mineral hydroxyapatite, organic material, and water, the dentin is not as strong as enamel and is yellow in color. If the enamel layer is damaged or faded, the yellow color of the dentin will be exposed, changing the color of the tooth. Dentin provides good support for enamel because it is less likely to be mineralized and more flexible than the outer layer. Dentin keeps forming throughout life while Unlike enamel, dentin can be generated throughout life to be healed naturally if the damage is not too much. This is a sensitive component of the teeth, so the patient may feel pain when the dentin is exposed to hot or cold temperatures or certain strong spices. Under the dentin is where the pulp is located. The pulp is the center of the teeth carrying connective tissue, blood vessels, and nerves. The pulp is the most vulnerable part of the tooth. Losing the protection of the enamel and dentin, the pulp is rapidly destroyed, which may cause a lot of pain because the nerves inside the pulp are damaged. The tooth hidden under the gum is protected by a strong layer similar to enamel but much thinner, less hard, and forms throughout its lifetime.

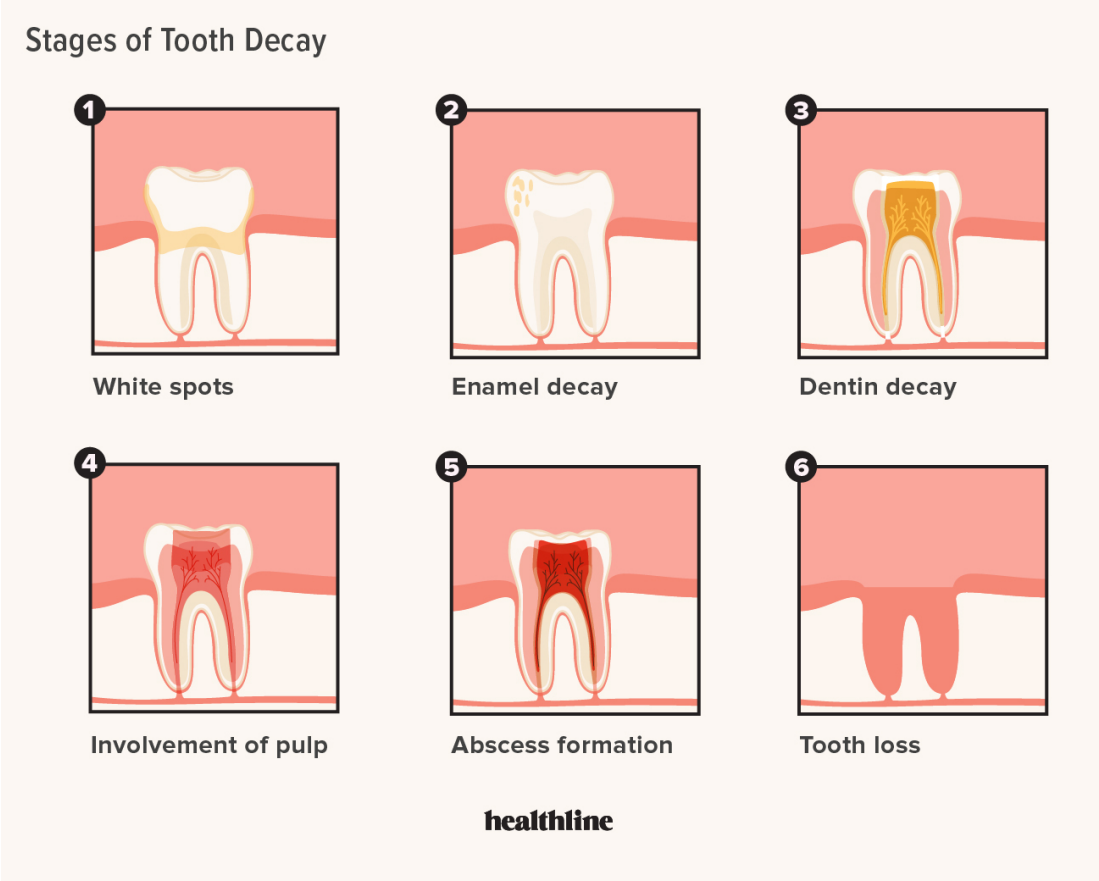



Figure 2.2: Different stages of tooth decay, image from [144]

Tooth decay happens when enamel (the outer surface of a tooth) is damaged by acids generated by bacteria living in the plaque inside a patient's mouth. According to [144] tooth decay can be found in one of the following stages. Firstly, a white spot can be seen as the enamel is slightly damaged for being attacked by acids. At this stage, the enamel can heal itself if the corrosion is stopped. However, if the problem is ignored, the layer will be further damaged, and the spot becomes darker and forms a hole in the tooth. These holes are called cavities or dental caries. Cavities cannot be healed automatically but need to be filled by professionals. After this stage, things start to get worse rapidly. If not treated, the tooth decay will reach the dentin, the weaker tissue protected by the enamel. The patient will experience pains when eating hot or cold foods because tubes that connect to nerves of the tooth are exposed. Afterward, if not being stopped, the damage will go further inside and reach the pulp, the inmost layer of the

tooth, where nerves and blood vessels can be found. The patient may experience some pains as it swells. The most serious tooth decay stage is when bacteria get inside the pulp, cause infection and generate pus there. This can get very painful not only at the tooth but the whole jaw and face. Because of the inflammation, the gums can swell, and the patient may experience fevers. At this stage, a dentist's treatment is essential to prevent from going into the bond and spreading to other areas. There are chances that the tooth is dead and must be removed. It takes a lot of effort to treat tooth decay when it gets serious, but one can prevent it from happening by practicing good dental hygiene.

### 2.1.2 Dental calculus

Figure 2.3: Dental calculus on the lingual of the mandibular anterior teeth, from [38]

Dental calculus (or dental tartar) is a strong plaque that covers some parts of teeth (Figure 2.3). It results from the chemical reaction happening when the minerals in saliva combine with gingival crevicular fluid (GCF) from dental plaque. This process kills bacteria inside the plaque but at the same time produces a form of strong precipitation with a rugged surface, inviting more plaque to build up. The tartar can develop on the gumline, go deeper inside and create the gap between gum and teeth. As a consequence, it weakens tooth roots and causes gum diseases. While the patient can remove dental plaque by brushing, it takes a dental professional to remove tartar when it is already hardened. Therefore, brushing teeth twice per day is a simple way to prevent dental calculus, and once the tartar is detected, the patient should seek dental care as soon as possible to ease the process of removing the tartar.

### 2.1.3 Gingivitis

Gingivitis (Figure 2.4): The early form of multiple periodontal diseases (PDs) affecting the gingiva and the supporting tissues of teeth. When being inflamed by gingivitis, the gums become swollen and red and bleed sometimes. Periodontal diseases are also the primary reason for tooth loss in adults. Research shows that the chance of having PDs increases when a person is under orthodontic treatment [172]. If not treated promptly, PDs can lead to serious diseases such as diabetes, pneumonia due to inhalation, strokes, and cardiovascular disease [128]. In contrast, the early stage of PDs, often known as gingivitis, can be treated much more easily than when it has progressed to severe levels. Slight gingivitis can be treated simply by brushing teeth correctly and regularly.

Figure 2.4: Gums with gingivitis (left, pointed by yellow arrows) versus healthy gum (right), from [183]

### 2.1.4 Gingival recession

Figure 2.5: Two teeth with gingival recession (pointed by the arrows), from [137]

Gingival recession (receding gums) is one of the most common PDs. It occurs when gingiva tissue is missing, exposing the teeth' root, as shown in Figure 2.5. The problem is mostly found in patients at the age of 40 or older, but sometimes it happens to teenagers [2]. There are several causes of gingival recession, such as PDs, dental tartar, aggressive tooth brushing, tooth crowding, hormonal changes, etc. Because the tooth root is not well covered, the tooth with receding gum becomes weak and over-sensitive to temperature and spices. It also creates a gap where bacteria can stay and directly attack the root of teeth. The consequences of receding gums can be mentioned as tooth displacement, leading to tooth loss, over-sensitive tooth, large holes between teeth, discolored tooth, and cavities at the tooth's exposed area. Taking care of dental health, including brushing teeth regularly and having checkups with dentists, is recommended to prevent a gingival recession.

### 2.1.5 Dental malocclusion

In the domain of dentistry, malocclusion refers to the misalignment of teeth. It negatively affects the function of teeth and causes multiple dental issues. Because teeth are not well aligned, it is difficult for them to brush their teeth properly, leading to cavities and gum issues. Most of the malocclusion cases are inherited, but some childhood habits may also lead to malocclusions such as overuse of pacifiers, frequent use of bottle feeding, or even thumb sucking [18]. Besides, there are a few cases in which the malocclusion is caused by sports injuries, poor dental care, or tumors. Most malocclusion conditions can be treated by orthodontic techniques, while some complicated cases may require jaw surgeries. Malocclusion can be categorized into five types:

Figure 2.6: An example of serious crowding of teeth, from [43]

- Crowding of teeth: (or crowded teeth) is a dental problem that happens when there is not enough room in the mouth for all teeth, as shown in Figure 2.6. It makes teeth bunch up, overlap, or even twist. Patients with crowded teeth usually have difficulties in oral hygiene, allowing the harmful bacteria to grow and lead to gum disease and tooth decay [65]. There are several levels of crowding which require different treatment techniques.

  In this research, we develop a machine learning model to recognize the crowding of teeth.

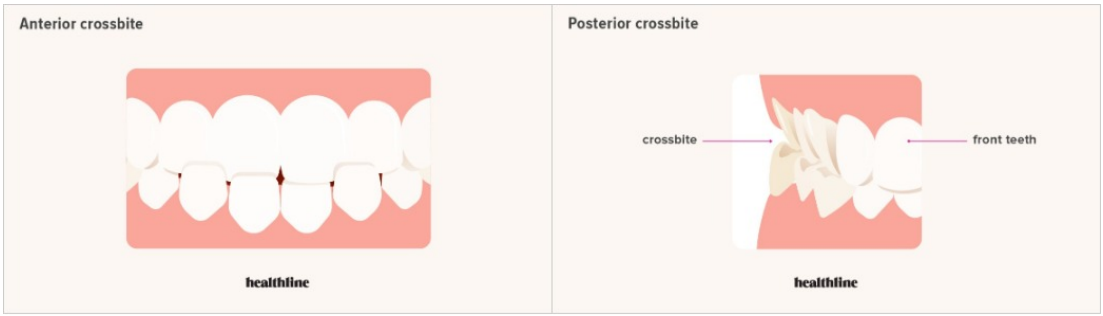



Figure 2.7: Two types of crossbite, from [174]

- Crossbite: is the condition that upper teeth and lower teeth cannot find each other when the person bites. It usually happens when a group of top teeth stays inside of lower teeth when biting. Crossbite can happen to front teeth (anterior crossbite) and molars (posterior crossbite) (Figure 2.7).
- Overbite: also known as buck teeth, refers to the condition that upper front teeth cover too much of the lower front teeth. This is usually a not serious problem, so the treatment is a matter of patient choice.
- Underbite: in opposite to overbite, means the upper front teeth do not go down enough when the person bites. It may cause difficulties biting, chewing, and speaking as well.
- Openbite: is the condition that the upper front teeth and lower front teeth do not touch when the person bites. People who have open-bite also have difficulties eating and talk. Compared to an underbite, an open-bite is more obvious, affecting the person's appearance and making them feel inferior.

As mentioned above, malocclusion, especially the crowding of teeth, makes it difficult to brush teeth properly. Insufficient dental hygiene leads to the accumulation of plaque and the formation of tartar and tooth decay. Tooth decay develops to cavities and slowly damages the tooth layer by layer while tartar builds up and attacks the gingiva by causing gingivitis and gingival recession. Therefore, treating malocclusion improves dental hygiene, leads to better dental health, and prevents a wide range of dental diseases. Orthodontic treatments can treat most malocclusion conditions [125].

## 2.2 Orthodontic treatments

Orthodontic treatment is the process that improves the appearance and performance of teeth correcting their alignment [44]. It may require removing or replacing teeth in certain cases. There are three main reasons for taking orthodontic treatments. First, it improves significantly the appearance of teeth which links to the jaws and face. Research has shown that the change in appearance after the orthodontic process has a markedly positive impact on the patient's mental health [78]. Specifically, post-orthodontic patients tend to be more confident in their appearance and participate more in social activities and sports. They become more positive, so anxiety and depression decreased. Second, it corrects the bite function. An unaligned bite can strain the jaw joints, which in the long run will cause jaw dysfunctions [168]. It is also the cause of wear on certain teeth' surfaces, making them more susceptible to acid and bacterial attacks. Third, bacteria are less likely to stick and multiply in the teeth after orthodontic treatment. The teeth, after being aligned, also make brushing easier. Thereby, the risk of oral diseases is significantly reduced, and oral health is improved.

This section describes the procedure of a usual orthodontic treatment and explains different techniques of the process.

### 2.2.1 Overview of the Process

The orthodontic treatment process includes a series of stages.

#### Oral Examination

First of all, the orthodontist performs a full exam of the patient teeth to evaluate the dental health and the complexity of the problem. Normally the patient needs to get panoramic X-rays to determine the position of teeth accurately. These scans show the exact biting position of all teeth and reveal if there is any tooth that is growing inside. The orthodontist sometimes requires a 3D scan to get more details that are not be clearly shown in the panoramic x-ray. The 3D scan is more costly than the panoramic x-ray, so it is required only when necessary.

### Treatment Planning

Based on the results of the test and the scans, the orthodontist creates a plan for the treatment, including the following factors:

- Oral health: Having good dental health is very important during the treatment because serious dental problems can cause interruptions during the process. Therefore, dentists need to ensure that the patient has good oral health. In some cases, if the patient has severe problems with the teeth or gums, the dentist will order these before starting the orthodontic process. If these diseases are mild, they can be treated in parallel with the orthodontic process.

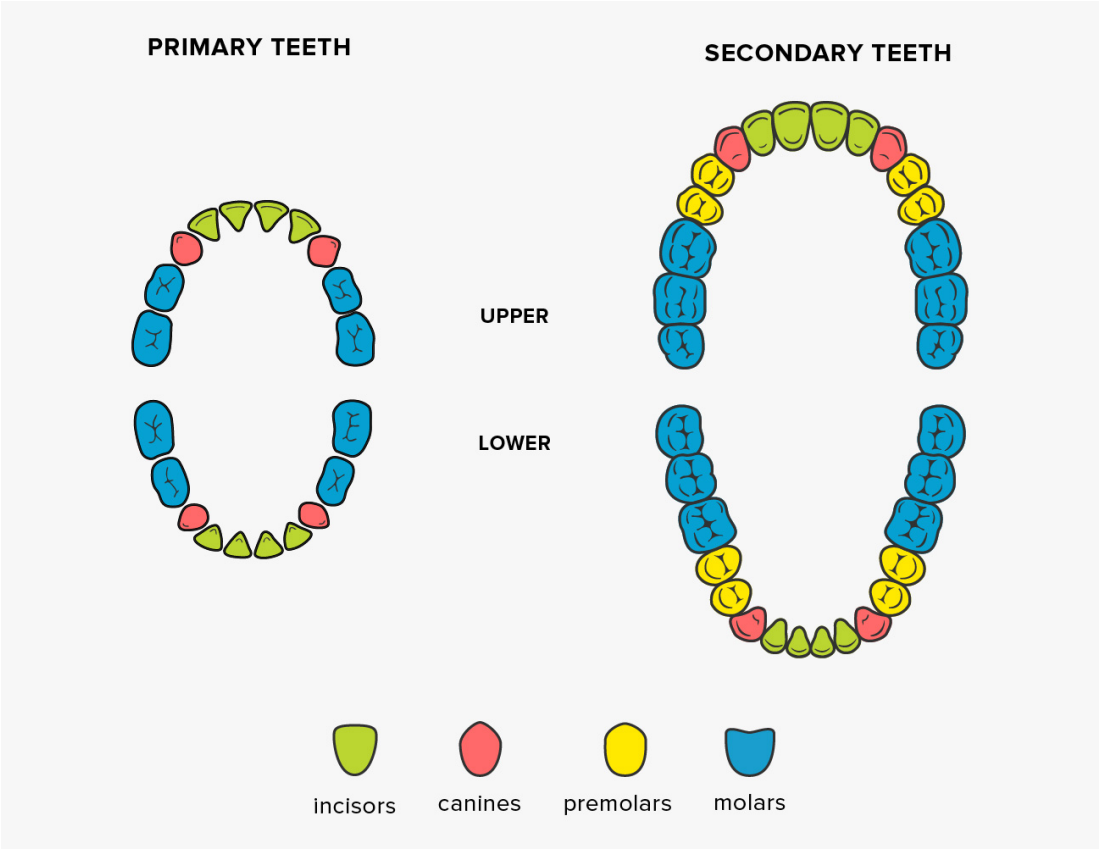



Figure 2.8: Types of human teeth, diagram from [175]

- Lower arch: Orthodontists often use the lower jaw as the starting point for planning movements because its size and shape are less likely to be moved than the upper jaw [136]. Sometimes, tooth extraction is required if the degree of crowding is too high. Premolars (bicuspids) are the teeth standing between cuspids and molars (see Figure 2.8). Working as transition teeth, they are usually removed in cases of crowding of teeth to gain spaces without affecting the appearance or function.

- Upper arch: Upper teeth are planned to fit around the lower ones so that they would make a proper bite. Extraction of a tooth in the lower jaw usually leads to the extraction of a corresponding upper one. Besides, the maxilla's premolars can always be removed if it lacks space for teeth to be aligned correctly.

- Choosing an appliance: Based on the movements being planned above (and teeth extractions if needed), the orthodontist needs to find out the appliance to achieve the goal. Normally, the same treatment outcome can be accomplished by different choices of appliances. Therefore, the orthodontist will propose all the possible appliances for the patient to choose. We will explain the most common orthodontic appliances in the next section.

#### Fitting appliance and checkups

After the appliance is placed, the patient will have a follow-up appointment with the dentist every 4-8 weeks. These checkups are intended to evaluate the orthodontic process. The dentist needs to make sure all teeth move according to plan. They will have to replace or adjust if any part of the appliance is damaged or not working properly. The patient's oral health is also carefully checked during each appointment as it directly affects the treatment's effectiveness. Each visit usually lasts an average of about 45 minutes.

#### Post-treatment

Depending on the case's complexity, the orthodontic treatment may last from a few months to a few years. At the end of the treatment, the appliance is removed, and a retainer is usually placed behind the teeth to prevent them from moving back to the initial unaligned state.

### 2.2.2 Orthodontic Appliances

During the orthodontic treatment, the patient has to wear appliances that put force on teeth to the right position. In general, the treatments can be done using different techniques, which vary in the level of convenience, duration, and cost. In this part, we describe the most common types of appliances and the corresponding components.

#### Metal Braces

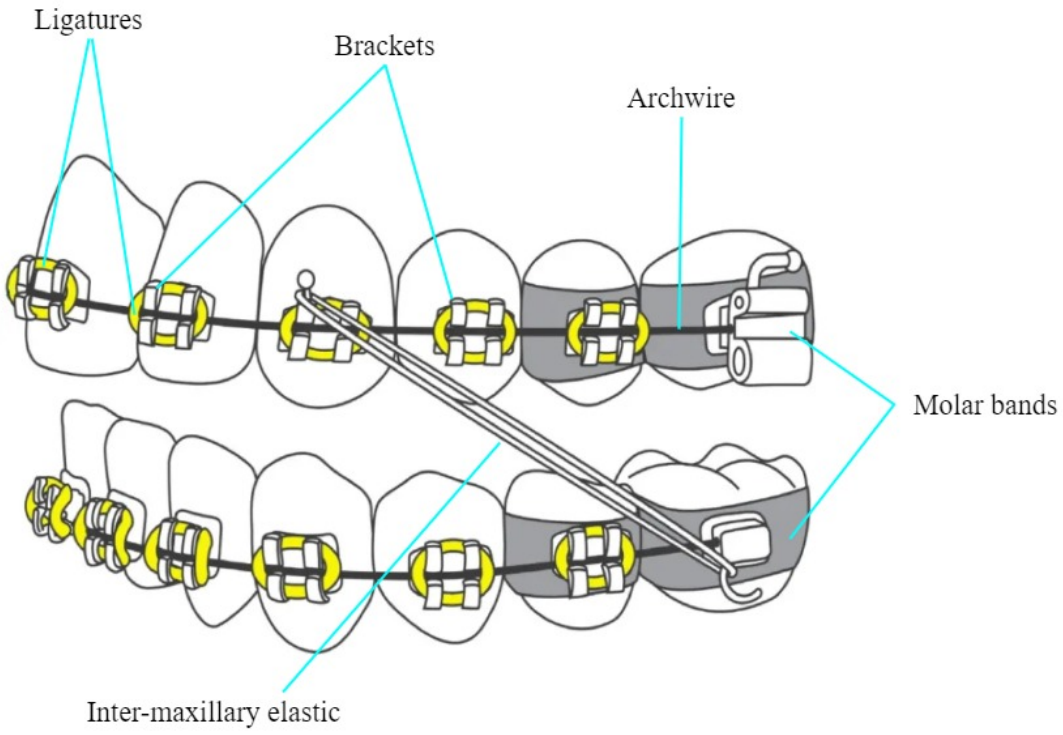



Figure 2.9: Metal braces with components, image from [6] with modifications.

Metal braces (also called traditional braces) are the most popular type of braces used for a wide range of dental malocclusion. Being shown in Figure 2.9, the technique consists of a few components: brackets, molar bands, archwires, either rubber bands or ligatures, and some auxiliary components such as power chains, coil springs, and inter-maxillary elastics.

- Brackets: are metal squares that are attached on the front-side of teeth using bonding cement. They are designed with a slot in the middle where the archwires go through and

control the patient's teeth' movement. Some brackets have a hook connected to them, which is used to attach auxiliary components such as rubber bands and coil springs.

- Molar bands: work similarly to brackets, but they wrap around the tooth. They are used in cases where it is difficult to use common brackets such as partially erupted teeth or not fully grown teeth. Orthodontists can use them to attach headgear as well.

- Archwires: are placed after brackets, and molar bands are already cemented. They move teeth via brackets and molar bands. Archwires can be made of several metal materials such as stainless steel or nickel-titanium.

- Ligatures: are used to fix the archwire with the bracket of each tooth firmly. They can be made of rubber bands with colors or twisted stainless steel.

Figure 2.10: Self-ligating brackets, from [16]

- Self-ligating brackets: (Figure 2.10) are special brackets that have their own ligatures so they can hold the wire firmly by themselves.

- Other auxiliary components: such as coil spring, inter-maxillary elastics, and power chain, can also be used for supporting the treatment with braces.

  - Coil spring: (Figure 2.11) is installed on the archwire to enlarge space between teeth, normally in the case of crowded teeth.
  - Inter-maxillary elastic: is another auxiliary that is attached to one tooth on the upper jaw and another from the lower jaw. It is used to make a force that controls the movement of teeth and sometimes for redirecting the jaws' movement.
  - Power chain: (Figure 2.12) is an elastic chain that is stretched over braces to reduce the spaces between teeth quickly. This is a preferred auxiliary for closing the space between more than two teeth.

Figure 2.11: Metal fixed appliances with rubber bands and coil spring (on the right), from [13]

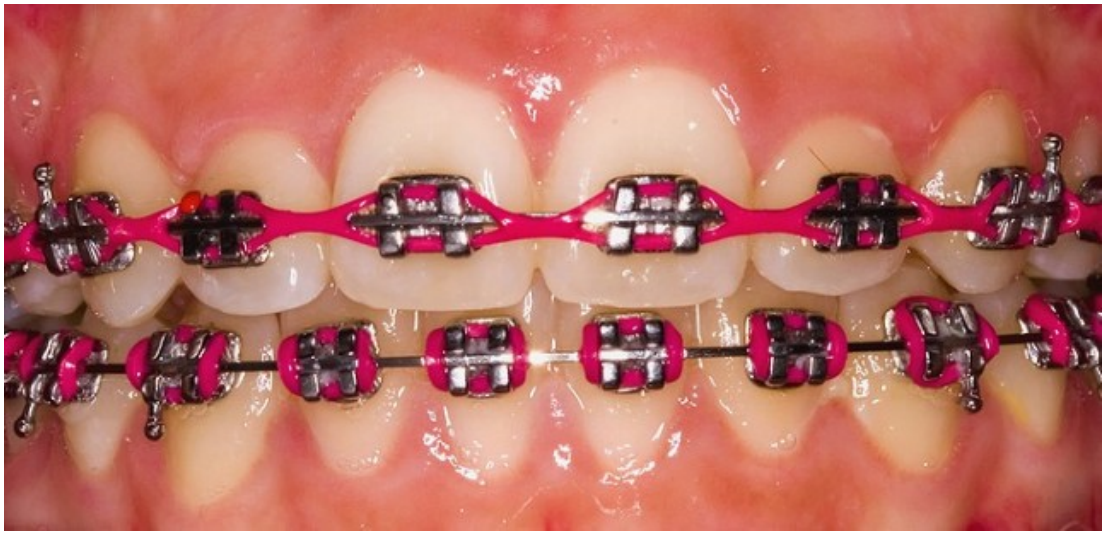


Figure 2.12: Power chain on the upper teeth and elastic rings on the lower teeth, from [126]

### Ceramic Braces

Ceramic braces work the same way as metal braces, but the brackets are made of ceramic which matches the color of the teeth [11]. Many patients choose ceramic braces because they are less visible than the metal ones so that their appearances do not change too much by wearing the appliances. The ceramic material's main advantage is that it has a similar color and texture to the enamel of teeth, making them less noticeable. They can customize the color of the brackets to match the color of the patient's teeth perfectly. Combining with white archwires and transparent ligatures, ceramic braces become even less visible.

However, ceramic braces also have some disadvantages. Firstly, they are approximately $30 - 40\%$ more expensive than metal braces. Secondly, they are less durable than metal and two times more likely to break during the treatment [22]. When a ceramic bracket is broken, it must be removed to install a new one. It increases the duration of the treatment and is harmful to the enamel surface of the tooth. Thirdly, ceramic brackets are larger than metal ones, so it is harder for the patient to brush their teeth, especially the gum line where plaque can build up, forming tartar and cause dental complications.

Figure 2.13: Ceramic braces with white archwires and white ligatures, from [11]

Figure 2.14: Lingual braces attached behind teeth, from [154]

### Lingual Braces

Lingual braces are another variant of traditional braces. Instead of attaching the bracket to the teeth' front-side, lingual brackets are fixed to behind the teeth to become invisible from the outside like in Figure 2.14. This method can be applied to correct most bite problems except deep overbite cases as the upper brackets can touch the lower teeth when the patient bites. Compared to the other two methods, lingual braces cause more difficulty when the wearer speaks because they limit the tongue's working space. Plus, it usually costs more than other treatment options. Therefore, since the invisible aligner is introduced, fewer people are wearing lingual braces [154].

### Clear Aligners

Clear aligners (or transparent aligners) is a different approach compared to the other three. The method uses transparent plastic trays as braces to apply a force on teeth to move them to the

Figure 2.15: An example of wearing transparent aligners on the lower teeth. Attachments are circled. Image from [21]

right position. The process is started by making a digital 3d model of the patient teeth. Based on this model, a computer program suggests a treatment plan including all the movements to be made and a series of aligners that gradually move teeth toward the designed direction. The patient must wear each aligner for about two weeks and at least 22 hours per day to work efficiently [110].

In many cases, attachments (or buttons) are also added to improve the aligners' performance. These attachments are small white dots of dental bonding that are attached to the surface of teeth. The way attachments and transparent aligners work are just like bracket and archwire in traditional braces. These attachments make the aligner stays firmly on teeth and provides anchor points so that the aligner can move teeth easily and precisely. Aligners can be combined with inter-maxillary elastics to correct the bite of the patient. To use the inter-maxillary elastics, additional buttons (usually made of metal) are attached to teeth at positions that are not covered by the aligner or directly on the aligner itself. Like brackets, these buttons and attachments are simply removed once the treatment is finished.

Clear aligners come with multiple advantages. Firstly, they are removable. Patients can remove aligners when necessary (maximum two hours per day) and put them back later. Brushing teeth without wearing appliances is also much easier and more efficient than when wearing braces. Therefore, during the treatment, patients wearing removable aligners have lower chances of having dental complications than traditional braces wearers. Secondly, these appliances are usually made of transparent plastic (Figure 2.15) that does not change the patient's appearance while wearing them. The smooth surface of aligners is also much more comfortable than metal or ceramic brackets, which may sometimes hurt the mouth's soft tissue. Thirdly, it requires fewer and shorter appointments with the orthodontist because the whole treatment is already planned, and all the aligners are produced at the beginning of the treatment. The orthodontist only has to make sure teeth are moving correctly so that the patient can switch to the next aligner.

Transparent aligners also have some drawbacks. The price of the treatment is usually higher than other options, except lingual braces. Those aligners are precisely printed using 3d-printers to match each patient's treatment stage while all components of other braces such as brackets, archwires, rubber bands, and so on are mass-produced. Moreover, letting the patient remove aligners when they want increases the chance that they may break or lose their aligners. It interrupts the treatment and makes it more expensive because they have to reproduce an aligner. Besides, if a patient loses her aligners and does not receive the replacement quickly enough, the teeth can move incorrectly, and the orthodontist has to make a new plan for the treatment again. For this reason, it is recommended to put aligners safely in their boxes.

In the following parts, we discuss in detail problems that can happen during orthodontic treatments and the importance of detecting them on time.

### 2.2.3 Appliances-related problems

In this part, we describe problems that may occur during orthodontic treatments related to appliances. In this thesis's scope, we focus on the two most popular appliances: braces and aligners.

Figure 2.16: An example of aligner not seated well on one tooth, from [40]

- Aligner: There are a few problems that may happen when a patient is wearing an aligner.
  - Incisal gap: is the space formed when a patient is wearing an aligner, but a tooth is not completely entering inside. It usually happens when a patient is not wearing the aligner correctly, or the movement of the teeth is not going as planned.
  - Absence of attachment or button: may happen when a great force is made on them, such as biting a hard food or brushing teeth carelessly. It is recommended to eat soft food for a few days at the beginning of the treatment because the bond between the attachments/buttons and the teeth is still weak, and they are more likely to fall off. Losing of attachment will reduce the aligner's performance, while missing a button will make it impossible to use the rubber band. Therefore, detecting the absence of attachments and buttons is essential to keep the treatment efficient.
  - Damaged aligner: This problem is less likely to happen, but the patient must receive a new aligner when it does. The patient themselves can spot the problem when the fragment damage is obvious, but sometimes, the damage can be so small that they do not notice. This is why an orthodontic monitoring system must take care of this problem too.

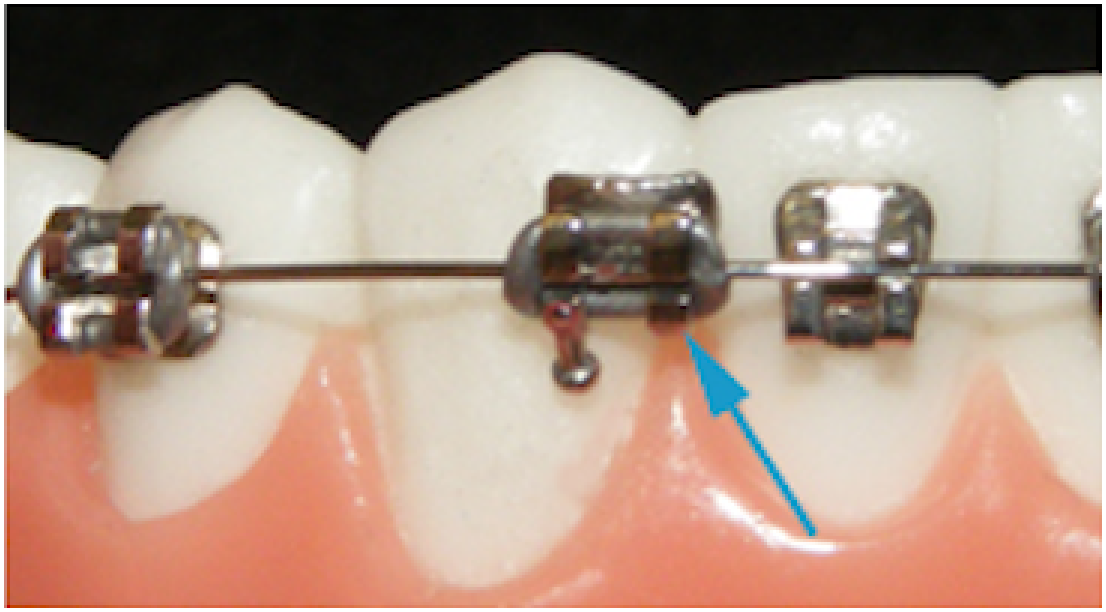


Figure 2.17: An example of a broken bracket which is detached from the tooth, from [135]

- Braces:
  - Bracket debonding: is usually performed by the orthodontist at the clinic to remove the bracket off the teeth once the treatment is done. However, for some reason, the bracket is detached from one or more teeth. This is a serious problem and must be informed to the orthodontist immediately for reparation. Most patients can realize it easily when it happens to one of the front teeth or when multiple brackets are detached at the same time, but sometimes patients do not notice the problem. Therefore, detecting debonding brackets should be detected by the dental monitoring system.
  - Loss of tie: is less emergence than the previous problem because sometimes the metal wire is still sitting in its place even when the rubber tie is lost. However, it is just a matter of time before the wire comes off the bracket because the presence of the tie strengthens the link between the wire and the bracket. Therefore, even though it is not urgent, losing ties should be detected and reported to the orthodontist.
  - Opening of the self-ligating clip: is as important as the loss of tie. The self-ligating clip has to stay closed to hold the wire in its place. Once it is accidentally opened, there is a high chance that the wire will get out of the bracket.
  - Power chain damaged and disengaged: are also monitored during orthodontic treatments. As it applies force on the teeth to close the gap between them, sometimes the power-chain can be broken. Losing or broken power chains will leave space between these teeth which negatively affects the treatment. There is a small possibility that the patient may swallow pieces of broken power chain with foods, which can be dangerous in certain situations.
  - Spring damaged or loosed: is less likely to happen to compare to the problems above, but it also can delay the treatment. Like other parts of braces, the coil spring can get loose or fall off when a strong force is applied to it.

In short, orthodontic appliances can have problems during the treatment, and each type of appliance has its own problems that can interrupt the treatment. Therefore, it is important for the patient to regularly visit the orthodontist to be detected and solved as quickly as possible if a problem happens.

## 2.3 Dental Checkups

Dental checkups, in general, are appointments between a patient and a dentist for evaluating dental health, recognizing dental problems, and proposing treatment if necessary. The dentist can make use of the appointment to remove plaque and tartar from the patient's teeth if they are already built up. Basically, a healthy person is recommended to visit the dentist once or twice a year, while a person with dental complications should be checked up as twice as frequently [176]. During orthodontic treatment, the patent needs to visit every one or two months.

In fact, people visit dentists much less often than needed. According to the survey in [155], more than one-third of French people visit their dentists less than once per year. One-fourth of them have never visited a dentist for any dental checkup in their lives. Similarly, about one-third of Americans visit their dentists less than once per year [180]. The main reasons that people are not getting dental checkups are their costs (59%), fear of dentists (22%), and the inconvenient time/location for the appointments (19%), as reported in [3]. Therefore, it is important to make dental checkups less costly and more accessible.

Online medical consultation (OMC) has recently become a promising consulting approach in multiple medical domains [108]. In OMC appointments, the patients do not meet the doctor in person but via a video call. OMC platforms usually make it possible for patients to talk with the doctor, transfer files and media such as photos and videos. It only requires an internet connection and a device with a camera connected to the internet. Because the appointments are not in-person, it is not limited by geography or weather conditions and flexible in time for both doctors and patients. As a result, online consultations are generally much cheaper than in-person consultations. One disadvantage of online consultations is that it is not applicable for certain checkups that require professional setup. Moreover, if the doctor wants to diagnose the patient, the diagnosis's result heavily depends on the quality of the images taken by the patient. For this reason, the process of taking photos should be simple enough so that patients can accomplish it correctly. It leads us to the next section, which explains the state-of-the-art online dental checkups and consulting platforms developed by the Dental Mind company.

## 2.4 Dental Mind

Dental Mind (initially H42, also known as Dental Monitoring, or DM) was founded in 2013 as the first company to provide a solution that uses smartphones to monitor teeth' movement during orthodontic treatments remotely. Today, the company has grown and provided three main services:

### 2.4.1 Dental Monitoring

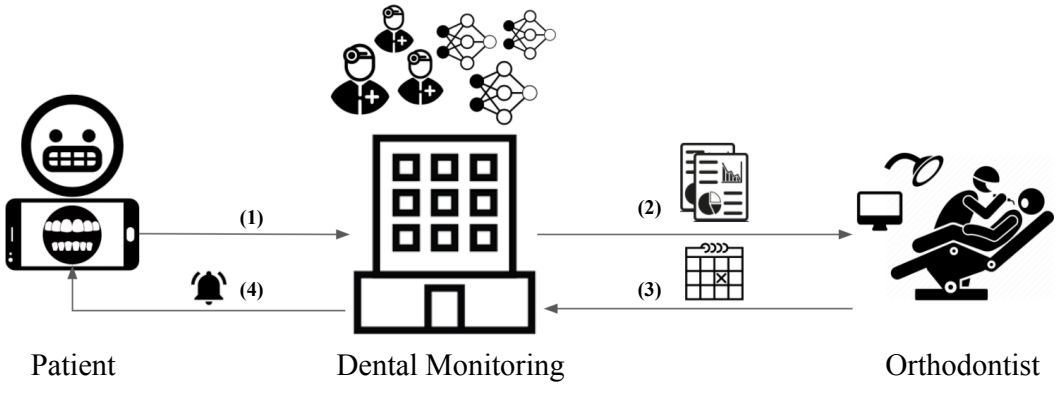



Figure 2.18: The working pipeline of Dental Monitoring service

Dental Monitoring is the initial service of Dental Mind. It enables orthodontists to remotely monitor the evolution of the treatment (including movement-tracking and problem detection). Thanks to the platform, they can also contact each other from far away. Figure 2.18 gives us an overview of the system. First, using the Dental Monitoring mobile application, the patient takes a series of photos of their teeth. The application also provides instructions step-by-step so that the patient can easily take pictures from all the required angles. Then, those photos are sent to DM to be processed and diagnosed by a team of dentists and dental experts assisted by computer programs, including scripted programs and learning-based programs. Next, a report is generated and set to the orthodontist. Based on the report, the orthodontist can monitor the evaluation of the treatment. They can send feedback and instructions to the patients and set an appointment if necessary. Alternatively, via Dental Monitoring, the orthodontist can define some rules, also called protocols, so that certain feedback will be sent automatically to the patients according to the report. We explain each steps in details below:

### Taking intraoral photos

The patient does this step with the help of the Dental Monitoring mobile application. There are two fashions for taking the scan: using only a cheek-retractor or using the DM ScanBox.

Figure 2.19: Wearing a too small cheek-retractor (left) and wearing a cheek-retractor correctly (right). Image from [31]

- Using a cheek-retractor: The scan can be done using a cheek-retractor and a smartphone with the Dental Monitoring application installed. The cheek-retractor is a piece of dental equipment that holds the patient's mouth open while retracting the lips and cheek to reveal as many teeth as possible. It is widely used in dental cliques for dental examinations and treatments. To start the scan, the patient puts on a cheek-retractor. It is important to wear the right size of cheek-retractor and wear it properly so that all teeth can be easily seen (Figure 2.19). Teeth will not show clearly if the patient wears a too-small cheek-retractor, while wearing an oversized one may cause pains. Following the application's instruction, the patient takes three sets of photos: closing teeth, slightly opened teeth, and widely opened teeth. For the first two series of photos, the scan is done from-ear-to-ear, so the patient slowly moves the camera horizontally (panning). The third series of photos focus on the molar teeth, so the patient must move the camera vertically. As the instruction is shown on the phone's screen, the patient must stand in front of a mirror.

- Using a DM ScanBox: This scan's equipment includes a cheek-retractor, a DM ScanBox, and a smartphone with a Dental Monitoring application installed. With the DM ScanBox, the patient does not have to worry about the camera's distance to their mouth or whether the retractor is well aligned with the phone. The box also stops noisy lights from the environment, which affect the quality of the photos. To start the scan, the patient launches the application and wears the cheek-retractor on. Then the smartphone is steadily placed in the front of the box with the camera facing inward. Afterwards, the patient lifts the DM ScanBox to the front of the retractor. When they are close enough, they will be

Figure 2.20: A person scanning her teeth using the DM ScanBox. Teeth are closed and the camera is moving horizontally. Image from [31]

pulled together automatically thanks to the magnets, and the scan is ready to be started. The camera movement in the scan using the DM ScanBox is the same as when using only the retractor: a horizontal scan with closed teeth, a horizontal scan with slightly opened teeth, and a vertical scan with widely opened teeth.

Once the photos are taken, the patient can preview the images before submitting them to the system.

### Image selection

Among the submitted images, there are usually duplicates and bad-quality images. Sent images are sometimes rotated because of the camera configuration of the smartphone. In this step, the images are cropped (and rotated if they are in the incorrect orientation), and the best photos are selected. From each submission, 10 photos corresponding to 10 required angles are chosen from over thirty submitted photos, as shown in Figure 2.21. Those 10 angles cover most of the visible sides of teeth, enables orthodontists and technicians to diagnose dental complications and detect problems. At the beginning of the thesis, Dental Mind already developed a few neural networks to handle this phase. One network is a classifier being trained to predict the camera's orientation of the given image. Based on the output of this classifier, the image is re-rotated to the usual orientation. Afterward, we apply a BLSTM network (see Section 3.2.2) to find out 10 best images for 10 best angles, because the mobile application usually takes multiple photos for each angle. The third network used in this step is an object detection model that draws a bounding box around the mouth region, the only valuable part for the following phases. Each image is cropped according to the predicted coordinate. Finally, these images are submitted to a queue, waiting to be diagnosed for dental complications and treatment-related problems.

### 3D Reconstruction

It is challenging to tell how much each tooth has moved by just looking at 2d images. Therefore, to be more precise, the technicians take the 3d model of the patient teeth (taken at the beginning of the treatment), rotate, move and scale teeth until the 3d model match the position of teeth from all perspective. By comparing this up-to-date 3d model with the previous ones, they can precisely compute how much each tooth has moved and realized whether the treatment is going

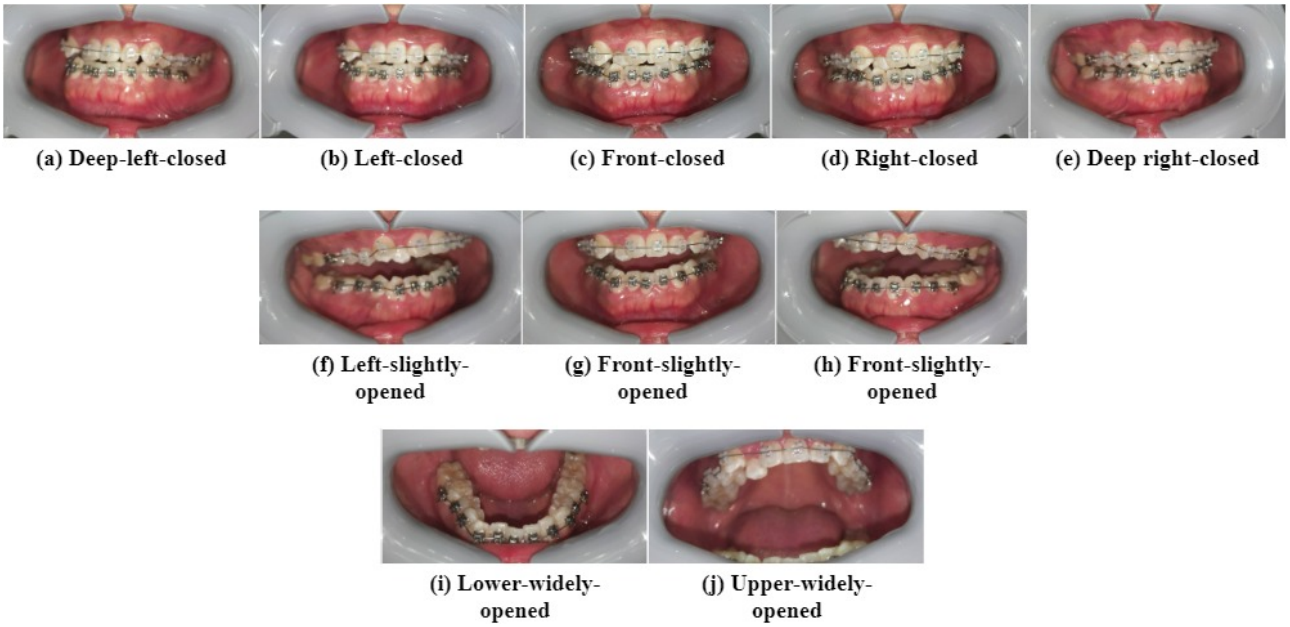



Figure 2.21: A photoset with 10 selected images after being cropped.

well or not. The orthodontist can access these insights via the Dental Monitoring dashboard, as shown in Figure 2.22.

### Diagnosis

Afterwards, the selected photos are diagnosed by technicians with the assistance of machine learning models. This step aims to find out as many common dental problems and orthodontic-related problems that it can see via photos as possible. Therefore, there are a handful of problems to be detected. Firstly, depends on the chosen treatment, appliance-related problems are checked. For example, if the patient is under treatment with invisible aligners, the technician will look for problems such as unseated aligner (incisal gap), detached aligner, aligner distortion, aligner fracture, absence of buttons, and absence of attachments. Otherwise, if the treatment is braces, they will look for braces-related problems. Most of these problems are already explained in Section 2.2.3. Then, they evaluate the intraoral health by reporting multiple problems such as dental calculus (Section 2.1.2), not sufficient cleaning of appliance, not sufficient oral hygiene, gingivitis (Section 2.1.3), gingival recession (Section 2.1.4), cavities (Section 2.1.1), detached dental crown and dental fracture.

Even though diagnosing dental problems via images only takes a few minutes, saving patients and dentists a lot of time, the burden is now shifted to the technicians who diagnose many dental images being submitted every day from around the world. The diagnosis must be accurate and quick, so they have to focus completely. As the task repeats, it becomes stressful. Therefore, it would be helpful to develop machine learning models that can handle simple predictions so that humans can focus more on complicated parts that take more dental experience and knowledge.

Another challenge of the Dental Monitoring diagnostic process is the uniformity of diagnostic results. To put it another way, different technicians sometimes come up with slightly different conclusions for the same picture. For example, when something looks like gingivitis appears on a photo, one technician can report immediately that gingivitis has occurred and suggest immediate treatment. At the same time, another would say that the sign is too mild and should be followed up for a more certain conclusion. However, if a learning model is well trained to recognize dental problems, the answers will always be consistent.

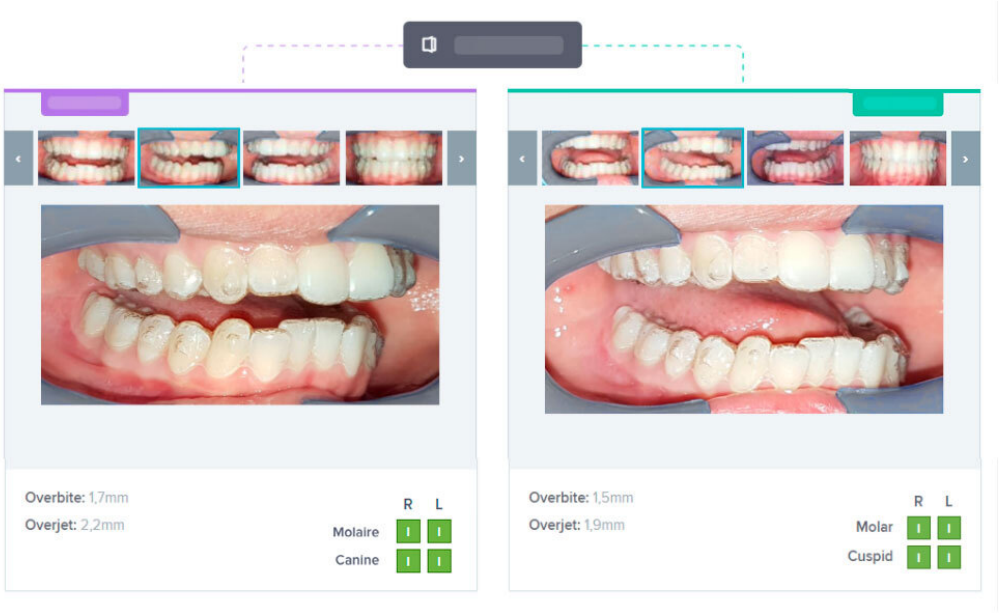



Figure 2.22: The interface of the Dental Monitoring dashboard where orthodontists can review and compare the diagnosis results of the submitted scans. Image from [32]

### Protocols

Via the Dental Monitoring platform, orthodontists can set up their protocols, in which they define instructions for technicians and validators in certain situations. For example, the orthodontist can design a protocol to automatically send a message to remind the patient to wear the aligner carefully if it is not well seated at certain teeth while ignoring the problem if it happens on other teeth which are not being treated. With protocols, orthodontists can also give some notes to technicians to pay more attention to some teeth or some certain problems regarding the ongoing treatment.

### Reports and Messages

When the diagnosis is made, a report containing chosen images and the diagnosis results is sent to the orthodontist. According to the protocol, a message is also sent to the patient with predefined instruction from the orthodontist. The patient can also see the images of the scan and a summarised version of the report. The patient can contact the orthodontist easily and send an unscheduled scan if they find it necessary through the mobile application. Messages between patients and orthodontists are delivered in real-time, which means patients can receive instant support when they need it. Communicating with the orthodontist between weeks of checkups also encourages the patient's commitment to the treatment.

## 2.4.2 SmileMate Virtual Consultation

SmileMate is an online consultation platform in which dentists/orthodontists can vision patients' dental health conditions. The service reduces the number of in-person appointments, which helps prevent the spread of Covid-19 [169] while enabling dentists to keep in contact with their patients. As a web-based service, it requires no application installation. SmileMate also makes it easier for new patients to accept treatment. In general, people do not visit dentists less than once per year, [3] which means that problems are detected when they have already started for months. With SmileMate, anyone can take a simple scan at home. The photos are diagnosed automatically, and the dentist can give recommendations. All of this is done with-

out an appointment. Knowing their dental problems and having advice from the dentist, the patient is now more likely to decide to get treatment. The way SmileMate works is explained in three steps below:

**Taking photos**

Figure 2.23: An example of photos to be taken on the SmileMate platform, image from [34]

Firstly, the patient visits the SmileMate website. Following the instructions, the patient takes a set of photos of their teeth. Unlike Dental Monitoring, SmileMate does not require a cheek-retractor for the scan because it serves new patients or people thinking about signing up for dental treatment. It is recommended to have a second person take the photos. The patient can use fingers to reveal the molars (Figure 2.23).

**Diagnosis**

Afterward, the photos are submitted to be diagnosed for dental problems like in the Dental Monitoring service. When the diagnosis is made, the dentist receives a report explaining the patient's oral health condition.

**Report and Recommendations**

The system can add recommendations to the report before being sent to the patient. The platform is considered a useful tool for orthodontists and dentists to lead new patients to be converted into acceptance of necessary treatments. Via SmileMate, dentists can also have online video consultations with their patients. It enables dentists to keep the contact with patients who have difficulties having regular physical checkups.

### 2.4.3 Vision

Showing patients how their teeth may look like during and after the treatment would improve the chance that a patient decides to take the orthodontic treatment and become committed during the process. The vision service includes two features: Appliance Selection and Smile Prediction. Advantage: personalized, using patient's teeth which makes it realistic.

**Appliance Selection**

Different types of orthodontic appliances have their own pros and cons. So each patient needs to choose the right appliance before starting a two-year-long treatment. One of the factors

that cannot be ignored when choosing a treatment method is appearance. Usually, the patient decides by looking at images of other patients wearing appliances. However, sometimes the reality does not match the imagination, and the patient may regret their decisions. Changing the type of appliance during treatment is not recommended as it is complicated and costly. Appliance Selection avoids this by virtually letting the patient "try" each appliance. It gives the patient a predicted vision of how their smile would look like during the treatment with each appliance type. At the moment of writing, the service covers the three most common types of appliance: metal braces, ceramic braces, and aligners, as can be seen in Figure 2.24.

(a) Input image (b) Output 1: Metal braces (c) Output 2: Ceramic braces (d) Output 3: Aligners

Figure 2.24: An example of images generated by Vision - Appliance Selection, images from [33]

The process of generating appliance images can be explained as follows. First, the orthodontist takes two images of the patient's smile: one with cheek-retractor and one without cheek-retractor via the Vision application installed. Then, the two images are submitted to the system to be edited by generative neural networks of Dental Mind. Finally, the results are sent back to the orthodontist. The whole process must not take longer than minutes because the patient should see the results right in the appointment so that the orthodontist can help the patient choose the most suitable appliance for them.

### Smile Prediction

It is reported that visualized goals are significantly more achievable than difficult-to-visualized ones [23]. When people can visualize their goal, they tend to be more motivated and committed to the goal. Orthodontic treatments are challenging as it takes time for teeth to move to the desired position. The patient also needs to take care of their teeth during the progress carefully. Therefore, having a predicted vision of the result will definitely motivate them to commit to the treatment. Smile Prediction works on the same input images taken for Appliance Selection, so Vision is actually a 2-in-1 service. Generative networks of the Dental Mind modify the photo of the patient's current smile to make it look more aligned. One advantage of this service is that it works on the given image, which means all the patient's teeth' characteristics are kept. It only aligns teeth and whitens them when being asked, As shown in Figure 2.25.

(a) Input image (b) Output image

Figure 2.25: An example of the image generated by Vision - Smile Prediction, images from [33]

# Chapter 3

# State of the Art

Convolutional neural networks (CNN), aka. deep neural networks, or deep learning, have constantly been growing in these years and achieved remarkable successes in many fields, especially computer vision [94].

In the field of medical imaging, researchers can apply deep learning to various parts of the human body. In [41], CNNs were trained for skin cancer detection on smartphone-captured images. The method archived comparable accuracy to the classification done by experts. In neuroscience, scientists applied deep learning to detect brain tumors on magnetic resonance (MR) images [60]. CNNs were also used for retinal vessel segmentation [48] from digital fundus photographs. Moreover, deep learning also helped to identify chest pathology [146] on a dataset of radiographs. Also, on x-ray photos, Spampinato et al. trained a deep network to assess skeletal bone age in human bone [152].

The development of deep learning also caused many new research directions in this field to be born. In this thesis, we focus on generative adversarial networks, image-to-image translation, domain adaptation techniques and try to apply them in the domain of orthodontics.

## 3.1 Object detection and instance segmentation

R-CNN [54] is known as one of the first works where convolutional neural networks (CNN) being applied for object detection. Using selective search [164], the method scans the image and generates about two thousand region proposals that potentially contain objects. Afterwards, these bounding boxes are resized and fed to a CNN for classification. R-CNN significantly outperforms conventional methods at the time, which are usually based on SIFT [105] and HOG [30] features. However, due to a large number of region proposals, the approach takes time for training and inference. Plus, the selective search algorithm in R-CNN is not trainable, which makes the model less adaptable. There are three main directions to develop the original technique.

The first way is to optimize the post-classification. He et al. propose SPPnet [64] with a spatial pyramid pooling (SPP) layer, which uses the same classifier for multiple input resolutions. The technique is robust to region size and scale and enables SPPnet to run noticeably faster than R-CNN. Fast R-CNN [53] uses a CNN for generating a feature map from the input image. This feature map is then used for identifying the region proposals, which are then wrapped, resized and fed into two fully connected (FC) layers that learn to predict the offset values of the bounding box and the category of the object inside. As a result, Fast R-CNN is almost ten times faster than R-CNN in training and about 20 times faster than R-CNN in testing.

Another direction of development is to use CNNs for improving the performance of the region proposals. Faster R-CNN [134] introduces a region proposal network (RPN) to predict the region proposals from the feature map. This is done with a training procedure that enables this network to share convolutional layers with Fast R-CNN. By doing this, the final classifier does not have to extract the mid-level features again but pool them directly from the output

of region proposals.

The third group of approaches does not include the proposals in their pipeline but directly predicts bounding boxes and categorizes them. Combining a deep convolutional network with the sliding window approach, OverFeat [145] computes the bounding boxes of objects from the classification feature maps. Because its localizer of OverFeat only gets local information when making a prediction, the method is more likely optimized for object localization than detection. YOLO [131] makes it even more straightforward by directly using the top feature map for predicting bounding boxes and the confidences of underlying object categories. This simplicity helps this method to perform real-time object detection.

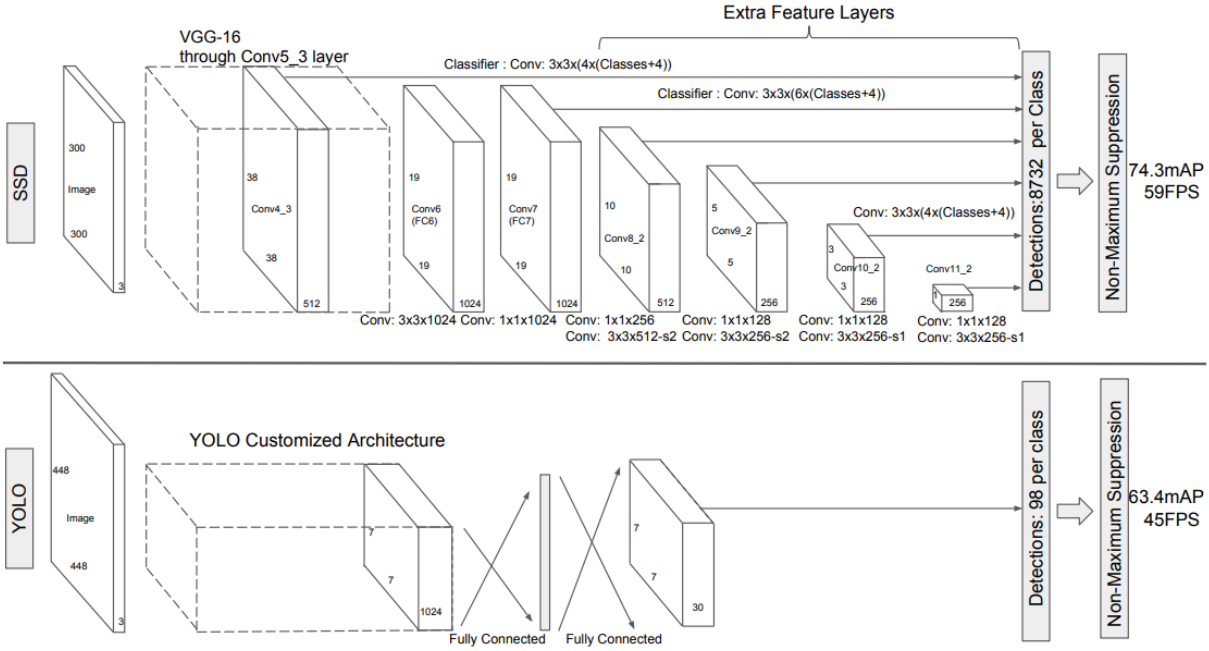



Figure 3.1: Comparison in the architecture of SSD and YOLO. Figure from [103]

Liu et al. propose Single Shot MultiBox Detector (SSD), [103] which uses multi-scale convolutional bounding boxes on multiple feature maps of the CNN (Figure 3.1). Like YOLO, SSD architecture takes the VGG-16 convolutional network [150] as its base, which predicts bounding boxes and classification scores of objects inside each box. Instead of operating detection on a single scale of feature map like in [145] and [131], the base network of SSD is improved with additional convolutional layers with sizes decreased progressively. Thanks to the feature maps of these layers, the prediction is made on multiple scales. These feature layers also replace fully connected layers in YOLO and make SSD run faster and more robust. Inheriting anchor boxes from Faster R-CNN, SSD applies them to multiple feature maps at multiple levels of resolution.

## 3.2 Recurrent Neural Networks

Recurrent Neural Networks (RNNs) are neural networks that have their output neurons connected and applied recursively to the inputs. The architecture performs very well on tasks that use sequential data such as natural language processing [55], signal processing [58], or text generation [158]. One main disadvantage of RNNs is the lack of time backpropagation which causes troubles when processing long sequences of data.

### 3.2.1 Long-Short Term Memory

Long-Short Term Memory (LSTM) [69] is the architecture proposed to address the problem of short-term memory in traditional RNNs. The method applies a memory & gating mechanism

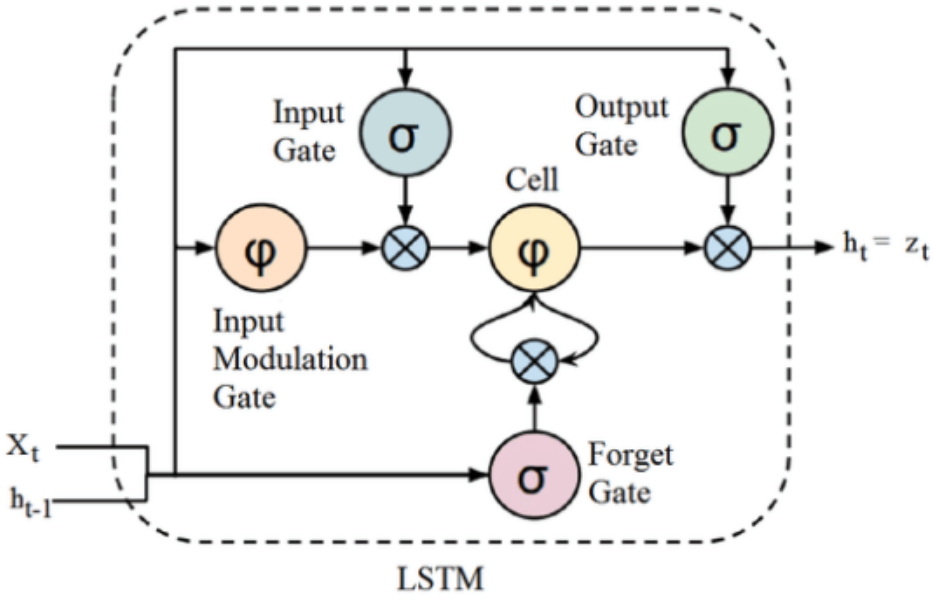



Figure 3.2: The architecture of a LSTM cell. Figure from [83]

Figure 3.2, to handle sequences containing long-term dependencies. The method performs very well in many applications such as handwriting recognition [19], music generation [111], and image captioning [179].

### 3.2.2 Bidirectional Long-Short Term Memory

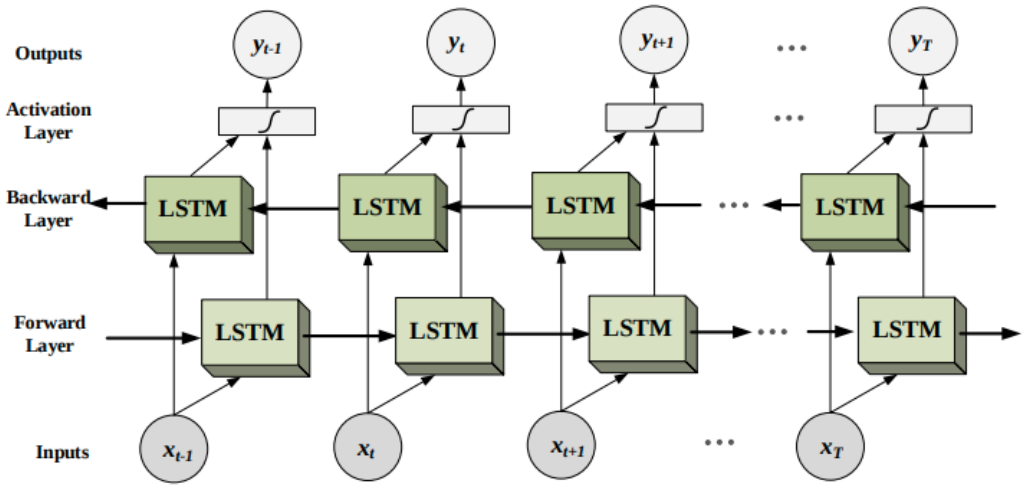



Figure 3.3: The overall architecture of BLSTM. Figure from [182]

A variation of RNN is bidirectional recurrent neural networks (BRNN) [141] which trains two recurrent networks at the same time, one for the forward direction and one for the backward. This architecture can access both states, before and after, of a sequential input at a time frame. This idea is also adopted in to LSTM networks to become bidirectional long-short term memory (BLSTM), in which, two LSTM networks are trained in two opposite directions of the sequential data as shown in Figure 3.3. The network is shown to outperform state-of-the-art models on sequence classification problems [57].

## 3.3 Adaptive Instance Normalization

Adaptive Instance Normalization (AdaIN) is a technique that normalizes the content features of an input image using parameters extracted from given style features. The method is widely used among generative networks and style transfer networks. The background of AdaIN includes batch normalization [79], instance normalization and conditional instance normalization.

### 3.3.1 Batch normalization

The original purpose of batch normalization (batch norm) is to stabilize and accelerate the training of deep neural networks by reducing the difference in the distribution of input between convolutional layers. One of the main challenges that one may face while training deep neural networks is the problem called internal covariate shift [79], which refers to the change in the distribution of layers' input during the training. This difference in distribution between features of layers in a convolutional network makes the update procedure unstable as it keeps aiming for a moving target. For this reason, deep networks take a lot of time to converge or cannot even converge at all. The idea behind batch normalization is to standardize inputs of the network's layers by fixing their means and variances. Batch normalization also makes the network less sensitive to the initial parameters and the choice of hyperparameters of the training. Therefore, it can be considered as a regularization technique besides dropout [153]. The method is found useful in both discriminative models and generative models [127].

Given an input batch $x \in \mathbb{R}^{N \times C \times H \times W}$ where $N, C, H$ and $W$ are respectively batch size, number of channels, height and width, the batch normalization of the batch $x$ is defined as:

$$BN(x) = \gamma \left( \frac{x - \mu(x)}{\sigma(x)} \right) + \beta. \tag{3.1}$$

During the training, $\gamma$ and $\beta$ are learned from the training data. In batch normalization, $\mu(x)$ and $\sigma(x)$ are the mean and the standard deviation of the input batch $x$. Note that for each feature channel, $\mu(x)$ and $\sigma(x)$ are computed independently across $N$, $H$ and $W$. Therefore, the mean and standard deviation of the feature channel $c$ of $x$ are respectively computed as:

$$\mu_c(x) = \frac{1}{NHW} \sum_{n=1}^{N} \sum_{h=1}^{H} \sum_{w=1}^{W} x_{nchw}, \tag{3.2}$$

$$\sigma_c(x) = \sqrt{\frac{1}{NHW} \sum_{n=1}^{N} \sum_{h=1}^{H} \sum_{w=1}^{W} (x_{nchw} - \mu_c(x))^2 + \epsilon}. \tag{3.3}$$

### 3.3.2 Instance Normalization

It is shown in [165] that replacing batch normalization layers in feed-forward style transfer networks with instance normalization layers can significantly improve their performance. The instance normalization is defined the same as batch normalization in Equation 3.1

$$IN(x) = \gamma \left( \frac{x - \mu(x)}{\sigma(x)} \right) + \beta, \tag{3.4}$$

except that the mean and standard deviation are computed independently, not only for each channel but also for each input sample:

$$\mu_{nc}(x) = \frac{1}{HW} \sum_{h=1}^{H} \sum_{w=1}^{W} x_{nchw}, \tag{3.5}$$

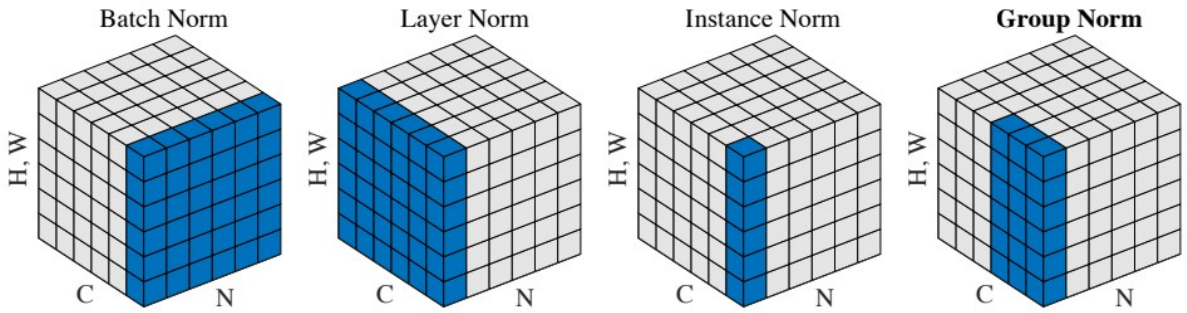



Figure 3.4: Illustration of normalization methods. Each cube represents a feature map tensor where $N$, $C$ and $(W, H)$ are respectively the number of samples in the batch, the number of feature channels and the spatial dimension. Figure from [177]

$$\sigma_{nc}(x) = \sqrt{\frac{1}{HW}\sum_{h=1}^{H}\sum_{w=1}^{W}(x_{nchw} - \mu_n c(x))^2 + \epsilon}. \tag{3.6}$$

Figure 3.4 visualizes the difference between different kinds of normalization methods, including batch norm and instance norm.

### 3.3.3 Conditional Instance Normalization

An extension of instance normalization is called conditional instance normalization (CIN) introduced in [39] for style transfer. For each style $s$, they propose to learn a set of parameters $\gamma^s$ and $\beta^s$ instead of learning one set of parameters $\gamma$ and $\beta$ in batch normalization and instance normalization. The conditional instance normalization is defined as:

$$CIN(x; s) = \gamma^s \left(\frac{x - \mu(x)}{\sigma(x)}\right) + \beta^s. \tag{3.7}$$

Thanks to CIN, a single convolutional network can learn to generate images in very different styles being learned from the training data. Experimental results in [39] show that the set of learned styles in CIN does not take many parameters of the network, but they completely change the style of images in the inference phase. The number of CIN parameters increases linearly with the number of styles and the number of feature maps, making it difficult to apply the method for large models and large datasets with tens of thousands of styles. Besides, the styles are learned only during the training phase, so the only way to apply new styles to the set is to retrain the network.

### 3.3.4 Adaptive Instance Normalization

While IN normalizes the input to a single style specified by the affine parameters, adaptive instance normalization (AdaIN) computes those parameters directly from a style input. Therefore, it has no learned parameters like $\gamma$ and $\beta$ in the previous works.

Given a content input $x$ and a style input $y$, the AdaIN layer is defined as:

$$AdaIN(x, y) = \sigma(y) \left(\frac{x - \mu(x)}{\sigma(x)}\right) + \mu(y). \tag{3.8}$$

In other words, after being normalized to get rid of the current style, the content input is scaled by $\sigma(y)$ and then shifted by $\mu(y)$ for applying the new style coming from the style input. These statistic components do not affect a lot the structure of the content input, but they make

changes in its style. They are both computed across spatial dimensions for each channel and each sample, like in instance normalization (Equation 3.5 and Equation 3.6).

## 3.4 Generative Adversarial Learning

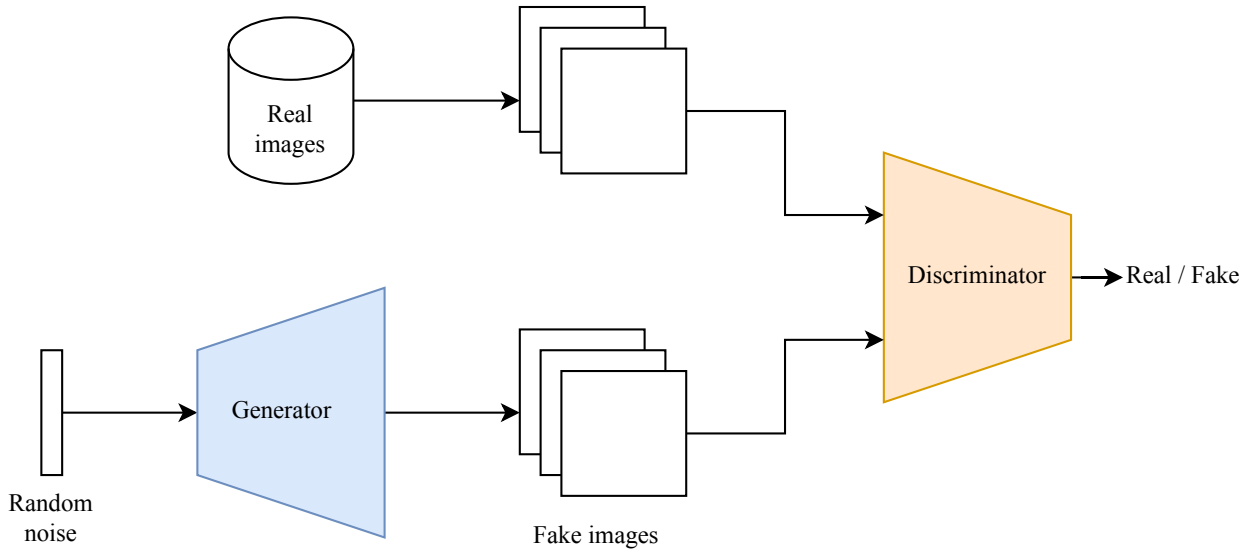



Figure 3.5: Generative adversarial network structure.

We will introduce original GANs, then some variants of GANs, especially the ones that lead to image-to-image translation solutions.

Generative adversarial nets were introduced in 2014 by Goodfellow et al. [56]. The GAN structure being shown in Figure 3.5 consists of two parts: a generator $G$, which learns the distribution of a given real dataset to generate plausible samples, and a discriminator $D$, which learns to distinguish between real and fake samples. The two models are trained at the same time and fight against each other in a mini-max game.

Let $x$ be a data point belonging to the distribution $p_{data}(x)$ and $z$ be a noise variable coming from a prior distribution $p_z(z)$. Generator $G(z, \theta_g)$ is a neural network that learns to map each data point from noise space to one in the data space. In the other hand, discriminator $D(x, \theta_d)$ is another neural network learning to label its input. The input of $D$ can be $x$ as the real sample or $G(z)$ as the generated sample. Overall, the objective function of GANs can be understood as below:

$$\min_G \max_D V(G, D) = \mathbb{E}_{x \sim p_{data}(x)}[logD(x)] + \mathbb{E}_{z \sim p_z(z)}[log(1 - D(G(z)))]. \tag{3.9}$$

The two networks are usually made of neural networks, but they can be other types of differentiable models too. One can imagine the generator as a criminal who tries to make fake money and the discriminator as a police officer who tries to tell if the money is real or fake. Both of them start the training simultaneously, so they are both bad at the beginning but get better after time. After a certain number of training steps, the discriminator should be very good at detecting fake images, while the generator should be able to generate real-looking images. GANs have been rapidly developed and adopted into many branches of research.

### 3.4.1 GANs Architectures

For improving the performance of GANs, multiple modifications in terms of architecture have been proposed. In [116], cGAN was proposed to make use of labeled data. The label is used as an additional input of the generator, encouraging the model to exploit attributes of the dataset.

The discriminator is also modified so that it does not only verify if the generated image looks realistic but also if it matches the given label. As a result, cGANs learn the difference between categories and are able to generate images associated with the chosen category. The drawback of this method is that it can only work on labeled datasets which are costly and not always possible.

In [132], GAWWN generates images with conditions in the form of text, bounding boxes and landmarks. InfoGANs [26] are introduced as a version of GANs, which can learn the disentangled representation of images without any annotations.

Another important extension of GANs is deep convolutional GANs (DCGANs) [127]. The work proposes a deep neural architecture for GANs, which comes with batch normalization [79], strided convolutions and ReLU to make the training efficient. DCGANs are further improved in [139] by using feature matching, a new objective function, which requires the generator to match the target data's statistics instead of directly maximizing the discriminator output. The authors also suggest using mini-batches for training the discriminator, historical averaging, label smoothing [159], and virtual batch normalization. The model succeeds in generating images 32 $\times$ 32 images from CIFAR-10 [89] but fails to generate objects when being trained with 128 $\times$ 128 images from the ILSVRC2012 [138] dataset with 1,000 categories. Salimans et al. [139] also introduce Inception Score as an evaluation metric for comparing GAN models.

Improving the resolution of the output is another important factor for developing GANs. ProGAN [85] learns to generate high-resolution images by progressively increase the number of convolutional layers in both generator and discriminator during the training. The idea is yet similar to StackGAN [184] and StackGAN++ [185], except for a few points. First, ProGAN does not take any additional input as a condition but directly generates images from noise. Second, early convolutional layers of ProGAN are kept for the whole training, while in we need multiple GANs to reach the high resolution. ProGAN succeeds in generating realistic 1024 $\times$ 1024 images. In [84], StyleGAN is proposed with significant upgrades, especially bilinear sampling and mapping network with adaptive instance normalization (AdaIN).

### 3.4.2 GAN Loss Functions

Realizing that the loss function of GANs is the key to stabilizing the training and the quality of the outputs, different modifications of the training objective are proposed. One disadvantage of the original GAN loss is that it saturates easily because it does not consider the distance of a data point to the decision boundary of the discriminator, but only the correctness of the predicted label. This is why the log loss is ineffective. Besides, when the gradient of the discriminator is saturating to 0, the generator will fail to learn as well because it is trained using the discriminator gradient.

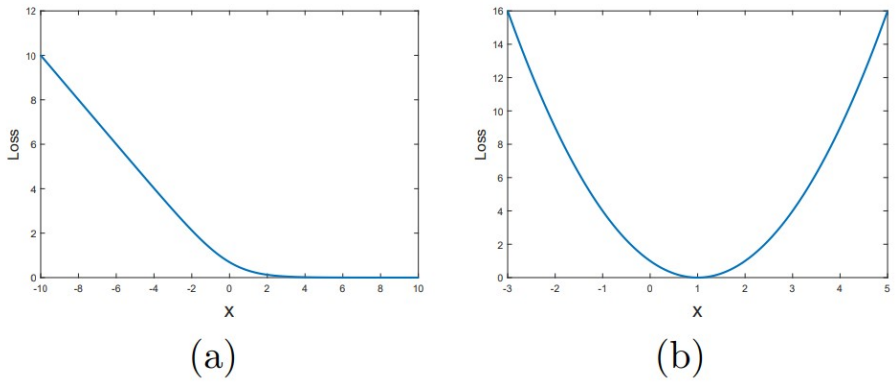



Figure 3.6: The sigmoid cross-entropy loss function in standard GANs (left) and the least-square loss function in LSGAN (right). Figure from [113].

Least-square GANs (LSGAN) [113], on the other hand, consider the distance of a data

point through the discriminator's prediction boundary. Specifically, it pushes data points that are far from the boundary toward the boundary by proportionally penalizes generated samples according to their distances to the boundary. This helps the generator slowly learn the real data distribution. Moreover, by penalizing the generated samples that stay far from the classification boundary, the loss function produces more gradients for training the generator. As a result, the training becomes more stable. Figure 3.6 visualizes the difference between the sigmoid cross-entropy loss and the least-square loss. While the sigmoid cross-entropy loss is flattened when the value of $x$ is large, the chance for the least-square loss to saturate is much lower. In contrast, LSGAN generates a great amount of gradient at these points. The objective functions of LSGANs is defined as:

$$\min_D V_{LSGAN}(D) = \frac{1}{2}\mathbb{E}_{x\sim p_{data}(x)}[(D(x)-1)^2] + \frac{1}{2}\mathbb{E}_{z\sim p_z(z)}[(D(G(z)))^2] \tag{3.10}$$

$$\min_G V_{LSGAN}(G) = \frac{1}{2}\mathbb{E}_{z\sim p_z(z)}[(D(G(z))-1)^2] \tag{3.11}$$

The method is widely used in early adversarial image-to-image translation methods such as [80] and CycleGAN [188]. Thanks to its stability, multiple networks that generate high-resolution and diverse outputs such as Pix2pixHD [173] and MUNIT [73] also adopt LSGAN for their training.

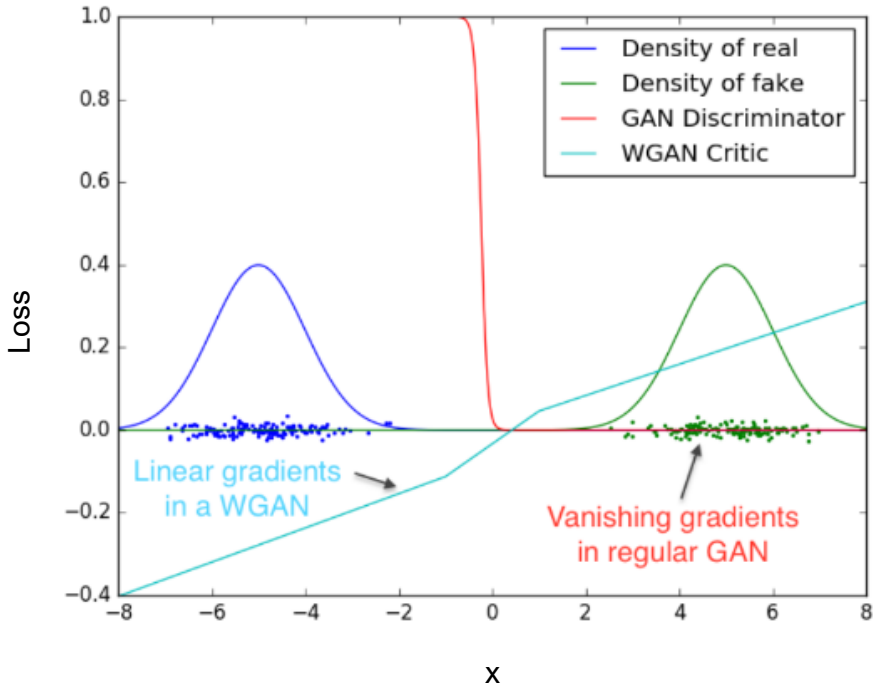



Figure 3.7: Comparison of an optimal discriminator and a critic when learning to differentiate two Gaussian distributions. WGAN critic always generates clean gradients while the discriminator saturates, leading to vanishing gradients. Figure from [5]

WGAN [5] is another alternative for the traditional GAN discriminator that applies Wasserstein distance in the training loss. In traditional GANs, the generator learns to minimize the gap between the real and predicted probability distributions for real and images that it generates. This objective is based on the idea of Kullback-Leibler divergence. WGAN, however, addresses the training of the generator from the point of view of the Earth-Mover's distance (EMD) or the Wasserstein metric. Generally speaking, given two distributions representing two ways of pilling up the same amount of dirt over a region, the EMD is the minimum effort of transforming one pile into another. The effort is computed by multiplying the amount of dirt being moved by the total distance of movement. Mathematically, let $\mathbb{P}_s$ and $\mathbb{P}_t$ respectively be

the source distribution and the target distribution, $\gamma(x, y)$ represents the amount of "mass" to be moved from $x$ to $y$ for transforming $\mathbb{P}_s$ into $\mathbb{P}_t$, and $\pi(\mathbb{P}_s, \mathbb{P}_t)$ represents the set of all joint distributions $\gamma(x, y)$. EMD is defined as:

$$W(\mathbb{P}_s, \mathbb{P}_t) = \inf_{\gamma \in \pi(\mathbb{P}_s, \mathbb{P}_t)} \mathbb{E}_{(x,y)\sim\gamma}[||x - y||]. \tag{3.12}$$

WGAN replaces the traditional discriminator predicting if an image is real or fake by a term called critic scoring how close an image is to the real data distribution. The advantage of the Wasserstein distance is that it is always continuous and differentiable even when the critic is already well trained, so the network does not saturate. Still, it keeps providing useful gradients that help to minimize the EMD, unlike a traditional discriminator, which tends to provide unreliable gradients once being well-trained 3.7. Also, being independent of the discriminator loss, WGAN is free from the problem of balancing a generator and a discriminator, making it easier to choose a network architecture. The training motivation is also eased to lower the generator loss instead of seeking an equilibrium between the two sub-networks. Therefore, the WGAN loss is related to the quality of the generated images leading to convergence toward the direction where the outputs are more realistic. This property is also helpful to understand the behavior of the model during the training.

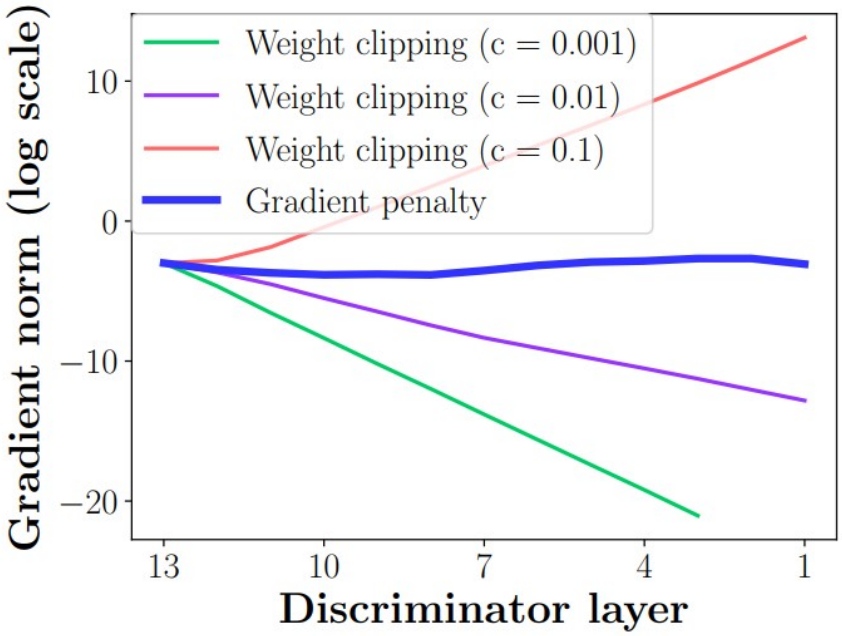



Figure 3.8: Gradient norms of WGAN critics (training on the Swiss Roll dataset [157]). A high value of $c$ causes the gradient to explode, while its low value leads to the vanish of the gradient. The gradient is stable when a gradient penalty is applied. Figure from [59]

In [59], it is pointed out that for achieving 1-Lipschitz functions, WGAN applies a technique called weight-clipping, which requires an additional parameter $c$. The research shows that the network performance is sensitive to the chosen value of $c$. To deal with the problem, Gulrajani [59] proposes an improvement for WGAN called gradient penalty (WGAN-GP) which is a soft version of the Lipschitz constraint in WGAN. In WGAN-GP, differentiable functions are 1-Lipschitz if and only if they have gradients of the norm at most 1 everywhere. The objective becomes:

$$L = \mathbb{E}_{\tilde{\mathbf{x}}\sim\mathbb{P}_g}\left[D\left(\tilde{\mathbf{x}}\right)\right] - \mathbb{E}_{\mathbf{x}\sim\mathbb{P}_r}\left[D\left(\mathbf{x}\right)\right] + \lambda\mathbb{E}_{\hat{\mathbf{x}}\sim\mathbb{P}_{\hat{\mathbf{x}}}}\left[\left(||\nabla_{\tilde{\mathbf{x}}}D\left(\tilde{\mathbf{x}}\right)||_2 - 1\right)^2\right], \tag{3.13}$$

where $\hat{\mathbf{x}} \sim \mathbb{P}_{\hat{\mathbf{x}}}$ are a random samples given during the training. The objective $L$ in Equation 3.13 consists of two elements the original critic loss of WGAN and the gradient penalty which is weighted by a parameter $\lambda$. As proposed in [59], the objective performs well with

$\lambda = 10$ on various network architectures and datasets. Besides, batch normalization is also removed for the norm of the critic's gradient is penalized for each input independently and because it does not contribute to the performance. As a result, the gradient of WGAN-GP is stabilized thanks to the gradient penalty as can be seen in Figure 3.8.

## 3.5 GANs Evaluation Metrics

The objective of GANs, and generative models in general, are to generate samples that match the given dataset. In this section, we explain several popular GANs evaluation metrics that are used in the thesis.

### 3.5.1 Inception Score

Inception score (IS) is one of the most popular metrics for evaluating image generation models' performance. Before describing IS, it is necessary to explain the deep neural network called Inception v3 [159]. The network is trained on the ImageNet dataset [138] for image classification tasks. ImageNet contains more than one million images that are divided into 1000 classes. The network learns to predict the category $y$ of each given image $x$, providing a vector of probabilities $p(y|x) \in [0, 1]^{1000}$.

Being trained on such a large and diverse dataset, Inception v3 can extract complicated features of images. Therefore, the model is often fine-tuned to implement image classification on target datasets (transfer learning).

The idea behind IS is to feed generated images to an Inception v3 network pre-trained on ImageNet and calculates a statistic of the output. Given $G$ as a generator that is trained to generate images $x$ from a latent distribution $p_g$, the inception score of $G$ is computed as:

$$IS(G) = \exp(\mathbb{E}_{x \sim p_g} D_{KL}(p(y|x)||p(y))), \tag{3.14}$$

where $D_{KL}(p||q)$ is the KL-divergence [167] between two given distributions. Basically, a high value of KL divergence refers to a large difference between $p$ and $q$. $p(y|x)$ is the conditional class distribution while $p(y)$ is the marginal class distribution. IS evaluates the generative network based on two factors. First, generated images must contain objects that are clear enough so that the classifier can confidently tell which class it belongs to, leading to a low entropy $p(y|x)$. Second, the generator must generate diverse images for all ImageNet classes, leading to a high entropy $p(y)$. If a generative model satisfies both factors, the KL-divergence of the two distributions will be large, which means IS will also be high. Moreover, poor quality images may still get high IS if they have clear local textures where deep convolution classifiers tend to pay attention to.

IS also has some drawbacks. In [10], it is pointed out that using the Inception v3 network, which is pre-trained on the ImageNet dataset, can be problematic. If a generative model is trained to produce images that are unfamiliar to ImageNet (Eg. dental images, skin cancer dataset, x-ray datasets, and so on), there will be a high chance that the Inception v3 network cannot classify the object, causing a low IS even though the output looks realistic. Besides, having high prediction confidence does not assure a realistic object generated in the image. Deep convolutional networks actually pay a lot of attention to local features for classifying images [17]. Therefore, if a GAN generates a non-sense image with many local features that give Inception v3 high prediction confidence, it can still achieve a high IS. Figure 3.9 shows some examples of unrealistic images which get almost perfect IS. In short, IS is not suitable for generative models, which are trained datasets that are different from ImageNet, and it can be tricked easily using local features.

Figure 3.9: Generated samples that achieve an Inception Score of 900.15 (the maximum value is 1000), which is much higher than state-of-the-art generative networks, which are on the order of 10. Images from [10].

### 3.5.2 Conditional Inception Score

Conditional inception score (CIS) is introduced in [73] as an extension of the original IS. The metric is designed especially for evaluating the performance of multimodal image-to-image translation models that generate multiple images from one single input image. It measures not only the quality of outputs but also the diversity of the outputs that are generated from one single input.

Considering the task is to learn the translation from domain $X_1$ to domain $X_2$. CIS requires that all the two domains' training samples are labeled based on their non-domain specified features. For example, if the two image domains are cats and dogs, they can be labeled according to the pose of the head. Assuming this requirement is satisfied, a classifier can be trained to predict the label of each sample in the two domains. The prediction is represented as $p(y|x_i), i \in 1, 2$. CIS requires a high entropy of $p(y_2, x_1)$ and a low entropy of $p(y_2, x_{1\to2})$ ($x_{1\to2}$ is the image transformed from domain $X_1$ to domain $X_2$). Combining the two objectives, we compute the CIS as:

$$CIS = \mathbb{E}_{x_1\sim p(x_1)}[\mathbb{E}_{x_{1\to2}\sim p(x_{2\to1}|x_1)}[KL(p(y_2|x_{1\to2})||p(y_2|x_1))]] \tag{3.15}$$

### 3.5.3 Learned Perceptual Image Patch Similarity (LPIPS)

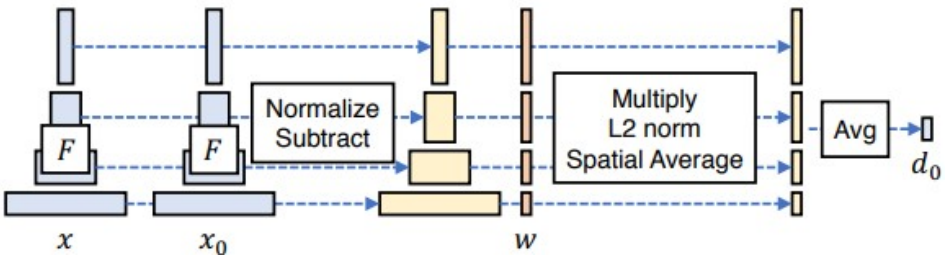



Figure 3.10: Computing the perceptual distance $d_0$ between two patches of images $x$ and $x_0$ using the LPIPS technique, diagram from [186].

LPIPS [186] is a deep learning based metric to compute the perceptual difference between pairs of images. In generative models, LPIPS is used to compute the diversity of the generated samples by computing the perceptual distance between all pairs. The method makes use of convolutional layers of deep networks, which are well-trained on large datasets for classification tasks to extract features from images, as shown in Figure 3.10. Given $x$, $x_0$ as two patches between which we want to compute the distance and the pre-trained network $F$, we firstly feeding the two patches to $F$ to get all the hidden features which are then unit-normalized in the channel dimensions. For the layer $l$ in $L$ layers of F, the normalized features of the two patches are expressed as $\hat{y}^l, \hat{y}^l_0 \in \mathbb{R}^{H_l \times W_l \times C_l}$. Then, after being scaled by a vector $w^l \in R^{C_l}$ and we compute the L2 distance of the scaled activations between the two patches. Computing the average spatially and the channel-wise sum, the obtain the final distance score. Mathematically, it is designated as:

$$d(x, x_0) = \sum_l \frac{1}{H_l W l} \sum_{h,w} ||w_l \cdot (\hat{y}^l_{hw} - \hat{y}^l_{0hw})||^2_2 \tag{3.16}$$

Compared to classical pixel-wise measures, LPIPS reportedly provides much closer answers to human judgment, which requires an understanding of high-order image structure and context. CNNs being used for LPIPS are usually AlexNet [90], VGG [150] and SqueezeNet [75].

### 3.5.4 Fréchet Inception Distance

Fréchet Inception Distance (FID) is another popular metric developed for evaluating the quality of images generated by GANs. The method is considered to be more consistent than IS [66]. As explained above, IS evaluates the quality of an image generator based on the classification performance of the Inception v3 network on the synthesized images. For this reason, IS actually measures how close the generated images to ImageNet, which is used to train Inception v3, instead of the target dataset. It is not assured to perform a fair evaluation on other datasets, especially the ones that are very different from ImageNet. One may think of training Inception v3 on the same dataset that is used for training GANs, but this solution is limited by the cost of labeling data and training the classifier. Once the dataset is modified, the classifier has to be retrained. To overcome this problem, FID compares the statistics of the fake image set to the statistics of the real dataset. Like IS, FID also uses the pre-trained Inception v3 model for extracting image features. Still, instead of getting features from all hidden layers like IS, it takes the output of the last pooling layer of the network to represent perceptual features of the image.

The value of FID is the Fréchet distance between two multivariate Gaussian distributions :

$$d^2((m, C), (m_w, C_w)) = ||m - m_w||^2_2 + Tr(C + C_w - 2(CC_w)^{1/2}). \tag{3.17}$$

## 3.6 Image-to-Image Translation

The translation of images from one domain to another has been a challenging problem in computer vision. Thanks to the evolution of convolutional neural networks, especially generative adversarial networks (GANs) [56], many deep learning models have been recently proposed to address the problem of image translation and achieve impressive outcomes.

### 3.6.1 Supervised Approaches

The Pix2pix method [80] is one of the earliest works on image-to-image translation based on conditional GANs. The generator $G$ in Pix2pix learns to map each image $s$ in the source domain (instead of one noisy input like the standard cGAN) to one image $x$ in the target domain. The discriminator $D$ is also different. It is trained to distinguish between the real pair $(s, x)$ and the

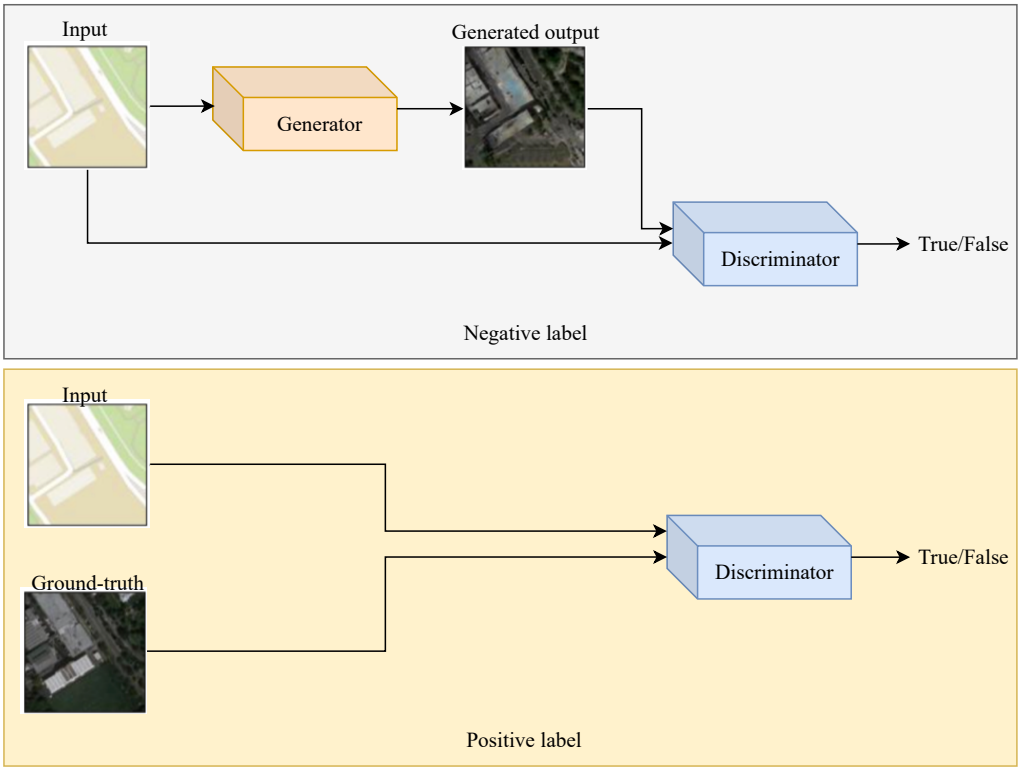



Figure 3.11: Pix2pix architecture.

fake pair $(s, G(s)$ As shown in 3.11. Pix2pix is trained in a supervised manner in a minimax game:

$$\min_G \max_D V(G, D) = \mathbb{E}_{(s,x)}[logD(s, x)] + \mathbb{E}_{(s)}[log(1 - D(s, G(s)))]. \tag{3.18}$$

It is able to generate images with a resolution up to 256×256.

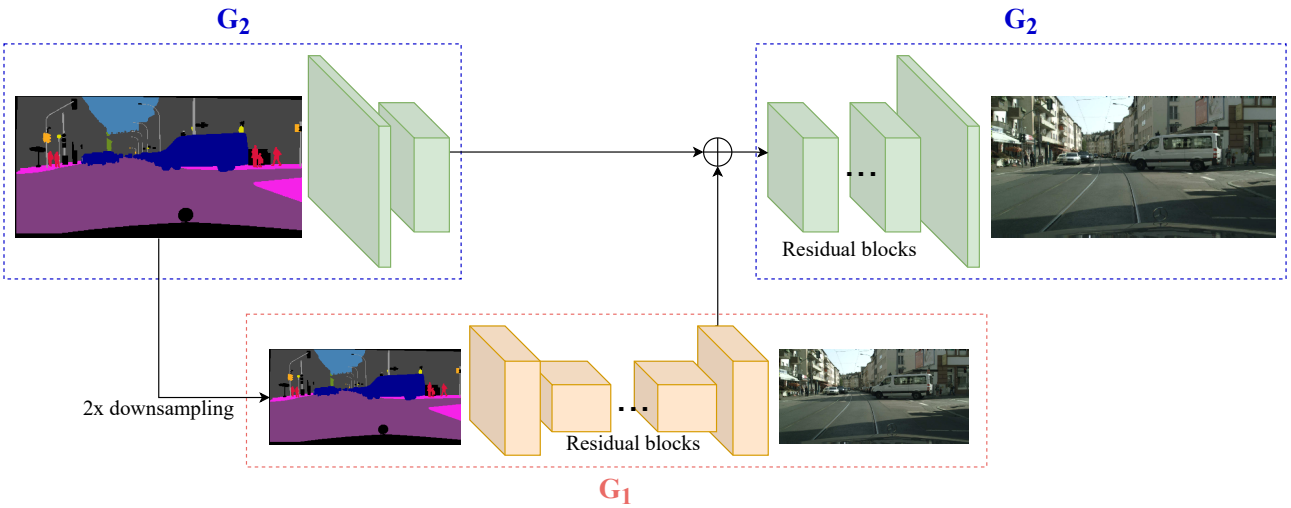



Figure 3.12: Pix2pixHD multi-resolution generator.

Wang et al. propose Pix2pixHD [173] as an upgraded version of Pix2pix with multiple factors to generate high-resolution images. Firstly, they introduce a coarse-to-fine generator demonstrated in Figure 3.12. The generator in the method represented as a tuple $G = \{G_1, G_2\}$. $G_1$ is the global generator, while $G2$ is called the local enhancer. $G_1$ consists of a convolutional layer, then a set of residual blocks and a transposed convolutional layer. The global generator takes a semantic map as input and returns an output image at the output, both of the resolutions

$1024 \times 512$. $G_2$ also consists of convolutional layers as a front-end, a set of residual blocks and the transposed convolutional layer as a back-end. The input image of resolution $2048 \times 1024$ is given to the front-end of $G_2$ while its down-sampled version ($1024 \times 512$) is given to the front-end of $G_1$. $G_1$ learns to generate an image according to the given segmentation map. Feature maps at the end of $G_1$ are element-wise summed with the feature maps generated by the front-end of $G_2$ to become the input of the residual blocks of $G_2$. Finally, these residual blocks and the back-end of $G_2$ learn to generate an image. During training, the global generator $G_1$ is trained before the local enhancer $G_2$. Then, the whole network is fine-tuned together.

To deal with multi-resolution generators, multi-scale discriminators are also introduced in the paper. Basically, they use one discriminator to evaluate the quality of the generated image at each resolution. These discriminators share the same structure but operate at different scales. The learning problem in Equation 3.18 becomes:

$$\min_{G} \max_{D_1,D_2} \sum_{k=1,2} V(G, D_k). \tag{3.19}$$

Furthermore, features extracted by convolutional layers of discriminators are also used to distinguish between real and synthesized images. This difference is call the feature matching loss. Let $D_k^{(i)}$ be the $i$th layer feature extractor of discriminator $D_k$, $N_i$ represents the number of elements in each layer, and $T$ the total number of layers, the feature matching loss is denoted as:

$$V_{FM}(G, D_k) = \mathbb{E}_{(s,x)} \sum_{i=1}^{T} \frac{1}{N_i} [||D_k^{(i)}(s, x) - D_k^{(i)}(s, G(s))||_1]. \tag{3.20}$$

Finally, the full objective of Pix2pixHD is the combination of the multi-scale GAN loss from Equation 3.19 and Equation 3.20, which is represented as:

$$\min_{G}((\max_{D_1,D_2} \sum_{k=1,2} V(G, D_k)) + \lambda \sum_{k=1,2} V_{FM}(G, D_k)) \tag{3.21}$$

where $\lambda$ is the weight controlling the importance of the two objectives.

### 3.6.2 Unsupervised Approaches

Learning to translate images using unpaired data is more challenging than with paired data because we do not know exactly which data-point in the source domain corresponds to which one in the target domain. Thus, it is reasonable to add some constraints to the training when it is possible. One popular assumption in most image-to-image translation research is that the structure of an image must not be changed too much by the translation. This is similar to language translation, in which a phrase must have the same meaning after being translated to another language. Shrivastava et al. [148] propose a training strategy in which a deep network learns to transform the style of synthesized images to make them look more real. To preserve the annotation, they add a pixel-wise loss between the style transfer network's input and output. Similar approaches are applied in later works, such as specific-task loss [15], semantic features [160], or distance between pairs of input samples [12] and so on. These constraints are useful for some specific tasks and datasets but cannot be applied robustly.

Cycle consistency is another well-known loss function being used in many bi-direction image translation models such as DualGAN [181], CycleGAN [188], and DiscoGAN [86]. In these networks, an image being translated from domain A to domain B can also be translated backward to obtain the original image and vice versa, as can be seen in Figure 3.13. The cycle-consistency loss is computed by the pixel-wise difference between the original image and the reconstructed one. Specifically, let $X$ and $Y$ be two domains that we want to learn the two mappings $G : X \rightarrow Y$ and $F : Y \rightarrow X$. The two data distributions are respectively represented as $x \sim p_{data}(x)$ and $y \sim p_{data}y$). The cycle-consistency loss is then computed as:

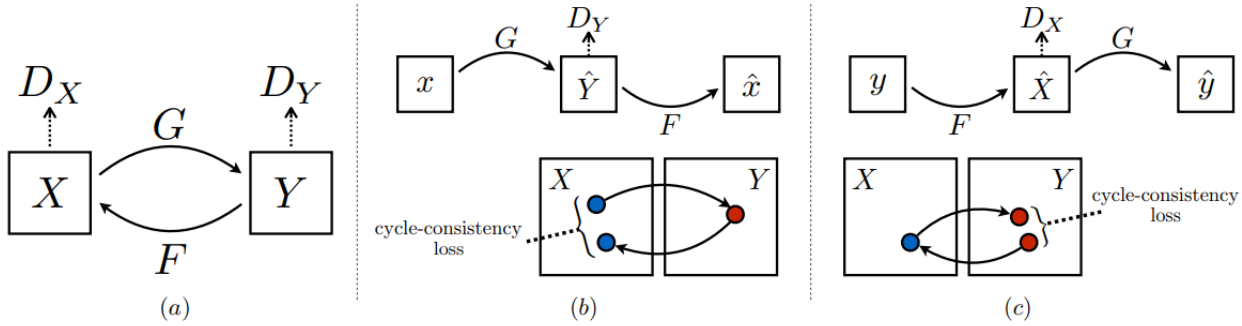
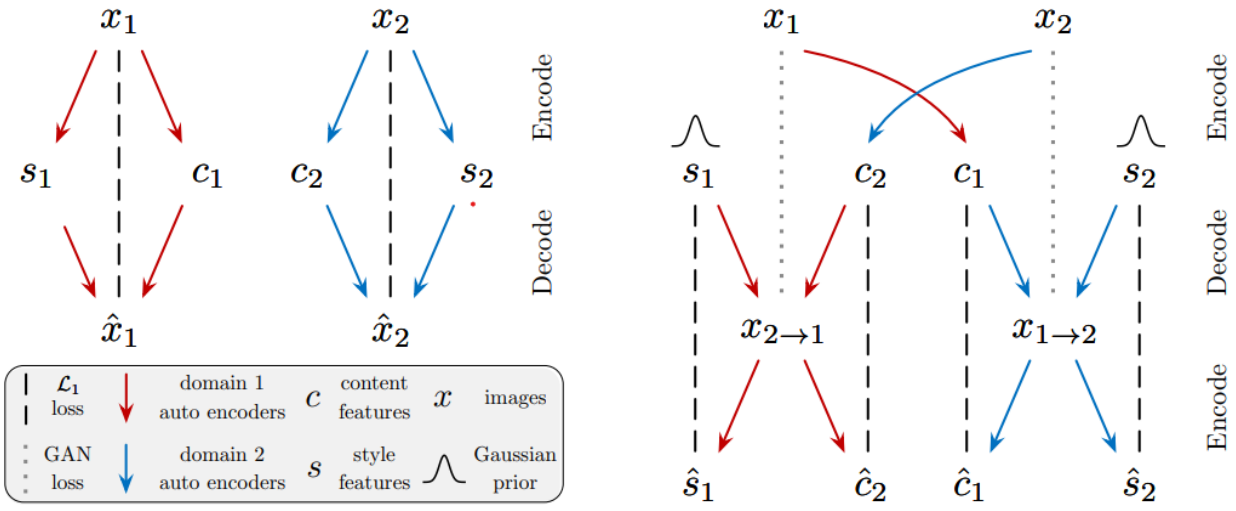



Figure 3.13: CycleGAN loses explanation: A pair of generators and a pair of discriminators for two domains (a). A forward cycle-consistency loss (b) and a backward cycle-consistency loss. Figure from [188]

$$\mathbb{E}_{x\sim p_{data}(x)}[||F(G(x)) - x||_1] + \mathbb{E}_{y\sim p_{data}(y)}[||G(F(y)) - y||_1] \tag{3.22}$$

As this cycle loss is not domain-related, it can be applied to most of the bi-direction translation models. In [1], Almahairi et al. extend CycleGAN for learning a many-to-many mapping by combining images with noises. Despite its ease of use, cycle loss does not assure any consistency in terms of annotation, which means labels of images can be flipped by the translation. Hoffman et al. [70] proposed to use both cycle consistency and semantic consistency during the training. However, this semantic constraint is not always accessible because it requires a pre-trained classifier of a similar dataset.

Another way to preserve the structural information after the transformation is to define a shared latent space where domain-independent features are stored. In UNIT [102], Liu et al. propose to break the translation into two stages: encoding the source image to a latent code and then decoding this code to an image in the target domain. In order to improve the diversity of the translation, some methods have been proposed to provide a fixed number of outputs [25] [52] [7]. In [187], Bicycle-GAN was able to generate multiple outputs from one input image, but this method requires paired data.




(a) Within-domain reconstruction

(b) Cross-domain translation

Figure 3.14: The overview of MUNIT. Figure from [73].

To gain additional control over the translated image features, Huang et al. develop MUNIT [73] as an extension of UNIT by splitting the latent code into two parts: content and style (Figure 3.14). Let $x_i \in X; i = 1, 2$ be the input image from each of the two domains. The

translation model in MUNIT consists of a content encoder $E_i^c$, a style encoder $E_i^s$ and a decoder $G_i$. An input image $x_i$ can be encoded into content code $c_i = E_i^c(x_i)$ and style code $s_i = E_i^s(x_i)$ while the decoder generates an image by combining a content code and a style code. For translating an image from one domain to another, eg. from $X_1$ to $X_2$, we first extract the content code $c_1 = E_1^c$. Then, we give $c_1$ to the decoder $G_2$ which generates image for the second domain. For the diversity of the output, $G_2$ takes $s_2$, a randomly sampled style code, as the second input. Therefore, the result of the translation is defined as:

$$x_{1\to 2} = G_2(c_1, s_2). \tag{3.23}$$

Similarly, we can have the opposite translation:

$$x_{2\to 1} = G_1(c_2, s_1), \tag{3.24}$$

where $c_2 = E_2^c$ and $s_1$ is also randomly generated.

The architecture of MUNIT is trained for multiple objectives, including bidirectional reconstruction losses and adversarial loss. The bidirectional reconstruction losses ensure that in each pair of encoder and decoder, they are inverses of each other. The bidirectional reconstruction objective can be understood that when an image is encoded into latent codes, and then these latent codes are decoded again, we should be able to obtain an image that is identical to the original one. Therefore we have two image reconstruction losses for two domain:

$$V_{recon}^{x_1} = \mathbb{E}_{x_1\sim p(x_1)}[||G_1(E_1^c(x_1), E_1^s(x_1)) - x_1||_1], \tag{3.25}$$

$$V_{recon}^{x_2} = \mathbb{E}_{x_2\sim p(x_2)}[||G_2(E_2^c(x_2), E_2^s(x_2)) - x_2||_1]. \tag{3.26}$$

Similarly, when a latent code is decoded to generate an image, then we should be able to reconstruct the latent code by encoding this image. Therefore, we have two content reconstruction losses and two style reconstruction losses:

$$V_{recon}^{c_1} = \mathbb{E}_{c_1\sim p(c_1), s_2\sim q(s_2)}[||E_2^c(G_2(c_1, s_2)) - c_1||_1], \tag{3.27}$$

$$V_{recon}^{c_2} = \mathbb{E}_{c_2\sim p(c_2), s_1\sim q(s_1)}[||E_1^c(G_2(c_1, s_2)) - c_1||_1], \tag{3.28}$$

$$V_{recon}^{s_1} = \mathbb{E}_{c_2\sim p(c_2), s_1\sim q(s_1)}[||E_1^c(G_2(c_1, s_2)) - s_1||_1], \tag{3.29}$$

$$V_{recon}^{s_2} = \mathbb{E}_{c_1\sim p(c_1), s_2\sim q(s_2)}[||E_2^s(G_2(c_1, s_2)) - s_2||_1], \tag{3.30}$$

where $q(s_i)$ is the Gaussian prior distribution, and $c_i = E_i^c(x_1)$ $(i = 1, 2)$ with $x_i$ is an input image from the domain $X_i$. Like many other image-to-image translation methods, MUNIT applies the $L_1$ function is to compute the element-wise differences between the original image/latent code and the reconstructed ones because it makes the generated images look sharper.

As a GANs-based bidirectional image-to-image translation method, MUNIT is also trained with an adversarial loss so that the generated images match the distribution of the target data. Given $D_i$ as the discriminator which distinguish generated images from $G_i$ and the images from the domain $X_i$, the adversarial losses are denoted as:

$$V_{adv}^{s_1} = \mathbb{E}_{c_2\sim p(c_2), s_1\sim q(s_1)}[log(1 - D_1(G_1(c_2, s_1)))] + \mathbb{E}_{x_1\sim p(x_1)}[logD_1(x_1)], \tag{3.31}$$

$$V_{adv}^{s_2} = \mathbb{E}_{c_1\sim p(c_1), s_2\sim q(s_2)}[log(1 - D_2(G_2(c_1, s_2)))] + \mathbb{E}_{x_2\sim p(x_2)}[logD_2(x_2)]. \tag{3.32}$$

The final objective of MUNIT is the combination of the losses above:

$$\min_{E_1,E_2,G_1,G_2} \max D_1, D_2 V(E_1, E_2, G_1, G_2, D_1, D_2) = \lambda_x(V^{x1}_{recon} + V^{x2}_{recon}) + \lambda_c(V^{c1}_{recon} + V^{c2}_{recon}) + \lambda_s(V^{s1}_{recon} + V^{s2}_{recon}) + V^{s1}_{adv} + V^{s2}_{adv}, \tag{3.33}$$

in which, $\lambda_x, \lambda_c$, and $\lambda_s$ are respectively the hyperparameters responsible for the weight of the reconstructions of images, content code and style code in the time the training.

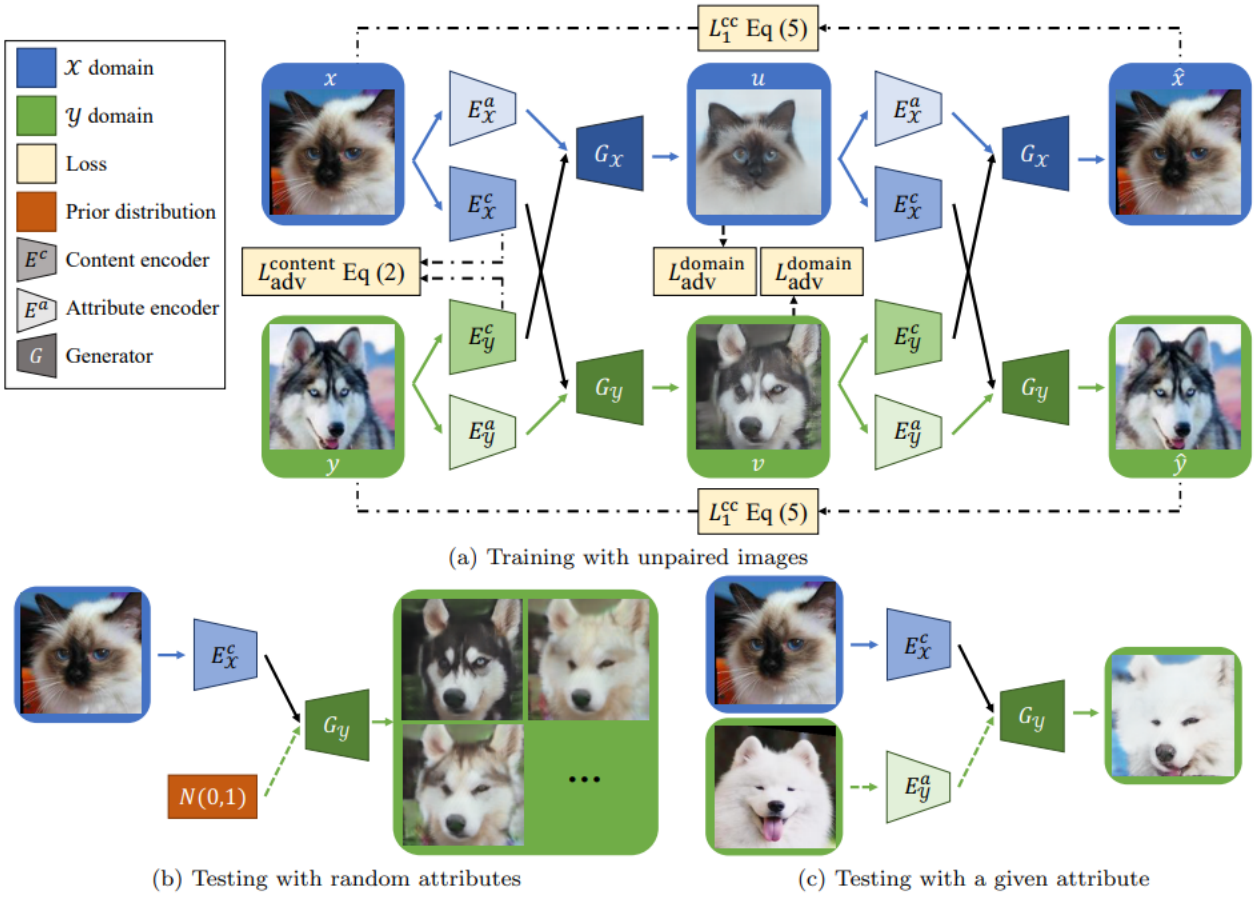



Figure 3.15: The overview of DRIT++. Figure from [97].

DRIT++ [97] is very similar idea to MUNIT with pairs of content encoders $E^c_x, E^c_y$, style (attributes) encoders $E^a_x, E^a_y$, generators $G_x, G_y$ and discriminators $D_x, D_y$ (Figure 3.15), where $x$ and $y$ are the two input images respectively from the two domains $X$ and $Y$. Moreover, DRIT++ also has a content discriminator $D^c_{adv}$ that distinguishes between the two domains' content codes. The goal is to trains the two content encoders to generate identical content codes to ensure they do not contain domain-related features, which means that there is only one content space for all the encoded content from the two domains. For this objective, a content adversarial loss is introduced:

$$\begin{aligned} V^c_{adv}(E^c_x, E^c_Y, D^c) = \mathbb{E}_x[\frac{1}{2} log D^c(E^c_x(x)) + \frac{1}{2} log(1 - D^c(E^c_c(x)))] \\ + \mathbb{E}_y[\frac{1}{2} log D^c(E^c_y(y)) + \frac{1}{2} log(1 - D^c(E^c_c(y)))] \end{aligned} \tag{3.34}$$

DRIT++ also applies a cross-cycle consistency loss, which says, when an image is transformed to another domain, with the content code extracted from the new image and the attribute of the original image, we should be able to reconstruct it. Specifically, let $x$ and $y$ are the two input images, $u = G_x(E^c_y(y), E^a_x(x))$ and $v = G_y(E^c_x(x), E^a_y(y))$ are the two cross-domain translated images, the cross-cycle consistency loss is explained as:

$$V^{cc}(G_x, G_y, E_x^c, E_y^c, E_x^a, E_y^a) = \mathbb{E}_{x,y}[||G_x(E_y^c(v), E_x^a(u)) - x||_1 + ||G_y(E_x^c(u), E_y^a(v)) - y||_1]. \tag{3.35}$$

Both MUNIT and DRIT++ store image style in an entangled manner.

### 3.6.3 Multi-domain Image-to-Image Translation

Most image-to-image translation methods are cross-domain models, which means they learn the mappings between only two domains using two generators. In reality, some problems require translations among more than two domains. For example, considering a face dataset with emotions labeled as "happy", "sad", "angry" and so on, we can learn mappings to change the emotion of the face. In this case, each emotion label can be understood as a domain. Let $n$ be the number of domains, then the number of generators that we need for mappings between all pairs is $n(n-1)$. Imagine we have a face dataset with 8 different emotions. We need to train 56 generators which is costly.

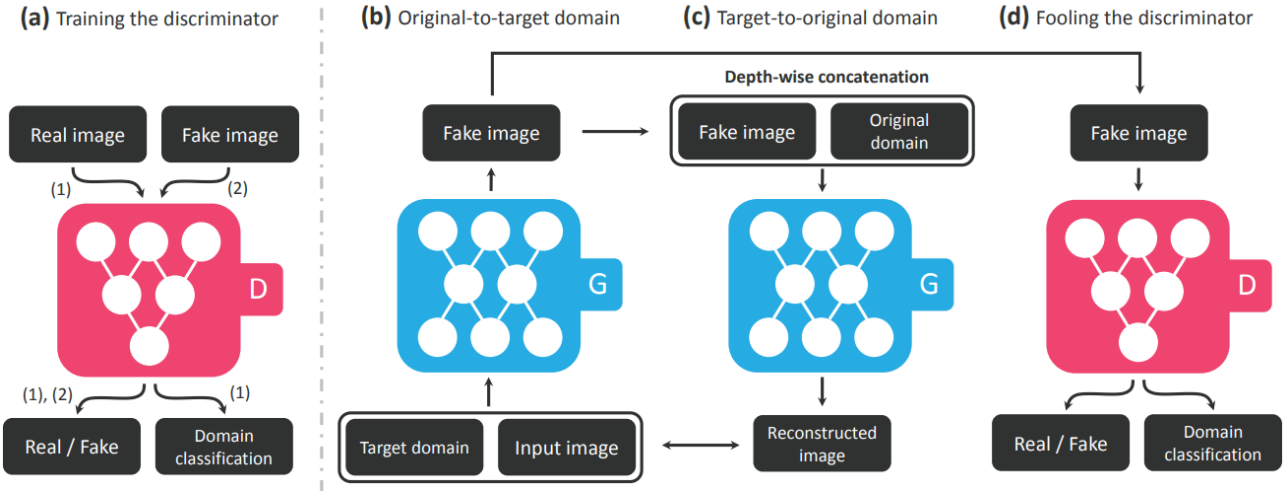
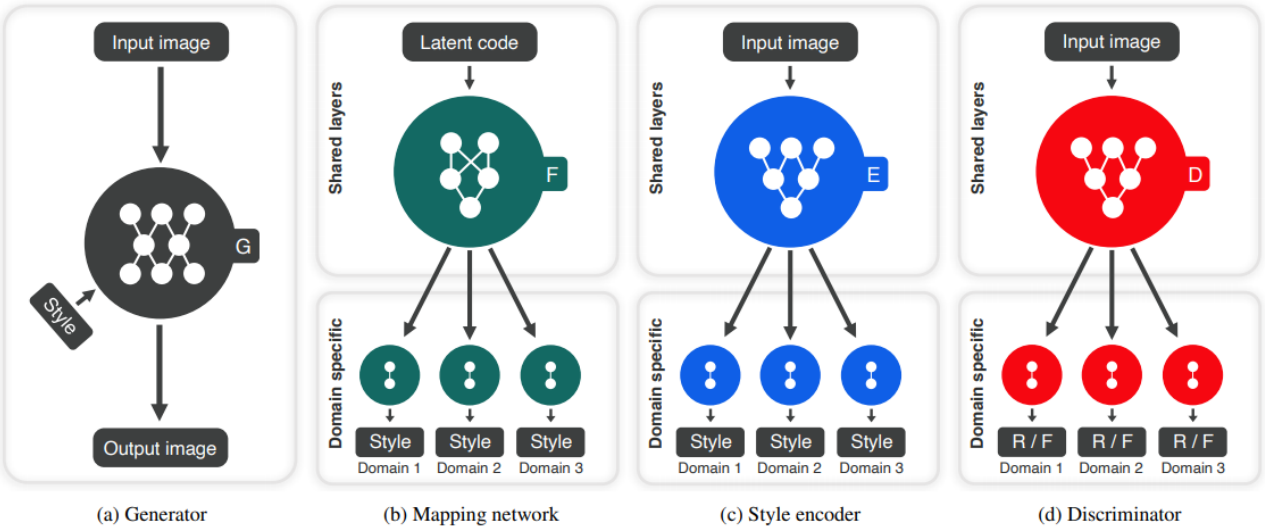



Figure 3.16: StarGAN architecture and training, diagram from [28]

In StarGAN [28], only one generator is trained to perform image-to-image translations for multiple domains. This approach has two main advantages. Firstly, it saves time and resources as only one model is trained for multiple domains. Secondly, the model becomes robust because it is trained using the data of multiple domains. Figure 3.16 demonstrates the architecture and the training of StarGAN. The discriminator $D$ of StarGAN does not only distinguish between real/generated images but also learns to predict the corresponding domain of images (Figure 3.16.a). Unlike in cross-domain image-to-image translation models where we have to build one discriminator for each domain, this single discriminator can work on multiple domains. The generator $G$ is also shared among domains, so it takes an input image $x$ and a label $c$ which defines the target domain (Figure 3.16.b). Using these two inputs, $G$ generates a fake image which is then sent to the discriminator (Figure 3.16.d). StarGAN also inherits the cycle-consistency loss from unsupervised former image-to-image translation models. Given the generated image and the label of the original model as inputs, $G$ is meant to generate an image that looks as close as possible to the original image.

StarGAN v2 [27] is introduced as a more robust version of StarGAN. The idea behind StarGAN v2 is that the style of a domain is diverse, so it should be represented as a domain-specific style code instead of just a single one-hot vector. The framework includes four components. In addition to a generator $G$ and a discriminator $D$, the network adopts a mapping network $F$ and a style encoder $E$, as shown in Figure 3.17. The generator (Figure 3.17.a) of StarGAN v2




(a) Generator (b) Mapping network (c) Style encoder (d) Discriminator

Figure 3.17: StarGAN v2 architecture and training, diagram from [27]

takes two inputs: an image $x$ from the source domain $X$ and style code $s$. The style code $s$ does not go through $G$ as $x$ but is injected into the generator's feature maps using a technique called adaptive instance normalization (AdaIN), which has become popular in GANs [73] [84]. Note that $s$ does not only contain the information of the target domain but also contains encoded additional domain-specific features that the generated image must contain. This style code can be generated by the mapping network $F$ or the style encoder $E$, which will be explained in the following. The mapping network (Figure 3.17.b) $F$ generates a style code $s = F_y(z)$ with $z$ and $y$ are respectively a random latent code and a domain label. $F$ starts with a multilayer perceptron (MLP) followed by multiple branches so that it can generate style codes for each domain differently. The structure of StarGAN v2 enables the mapping network to learns the attributes of images represented as domain-specific style codes. The style encoder $E$ (Figure 3.17.c) also generates a style code corresponding to each target domain, but unlike $F$, it extracts the information from a given image $x$. Giving the encoded style to the generator $G$, we transfer the style from a reference image to another. This is a significant advantage of StarGAN v2 to the first work. Like the other sub-network of the architecture, the discriminator $D$ (Figure 3.17.d) also has domain-specific branches (denoted as $D_y$) that learn to distinguish between real and fake images for each domain.

StarGAN v2 is trained for multiple objectives. Let $x \in X$ be an input image and $y \in Y$ the original domain of $x$. A latent code $z \in Z$ and a target domain $\tilde{y}$ are randomly sampled during the training. The mapping network $F$ generates a domain specific-style code $\tilde{s} = F_{\tilde{y}(z)}$ which is then given to the generator $G$ which produces an output image $G(x, \tilde{s})$. The adversarial loss is then denoted by:

$$V_{adv} = E_{x,y}[logD_y(x)] + E_{x,\tilde{y},z}[log(1 - D_{\tilde{y}}(G(x, \tilde{s})))]. \tag{3.36}$$

For minimizing the adversarial loss, $F$ must learn to generate the style code $z$ that corresponds to the given $y$. In contrast, $G$ must learn to provide images that $D$ would fail to recognize as a generated image.

The method also has a style reconstruction loss so that the generator $G$ must learn to include features in the style code $\tilde{s}$ into the generated image $G(x, \tilde{s}$. We do this by giving the image to the style encoder $E$ to get an extracted style code and minimize the difference between this reconstructed code and the style code $\tilde{s}$. The style reconstruction loss is denoted as:

$$V_{style} = E_{x,\tilde{y},z}[||\tilde{s} - E_{\tilde{y}}(G(x, \tilde{s}))||_1]. \tag{3.37}$$

Minimizing this loss forces $G$ to generate images using the giving style code instead of randomly generating samples as in the traditional GANs. We can find a similar style-reconstruction

loss in multiple image-to-image translation techniques [102] [73] that encode the style of the input image into a latent code. However, StarGAN v2 uses only one style encoder for multiple domains. As mentioned above, an advantage of StarGAN v2 compared to the original StarGAN is generating diverse images. The generator $G$ is enforced to maximize the difference between images that are generated using the same input image but different style codes. Specifically, the diversity sensitive loss is denoted as:

$$V_{diverse} = E_{x,\tilde{y},z_1,z_2}[||G(x,\tilde{s_1}) - G(x,\tilde{s_2})||_1], \tag{3.38}$$

where the style codes $\tilde{s_1}$ and $\tilde{s_2}$ are respectively $F_{\tilde{y}}(z_1)$ and $F_{\tilde{y}}(z_2)$. By regularizing $G$ with this term, we encourage the generator to explore as much as possible the dataset distribution so that it can generate diverse images containing meaningful features. Last but not least, the widely-used cycle-consistency loss is also included in StarGAN v2. It ensures that the generator $G$ only modifies domain-related characteristics of the input image while keeping other features on the generated image. In detail, given an input image $x$, we have $y$ as the original domain, and $\hat{s} = E_y(x)$ as the extracted style code of $x$. To satisfy the cycle-consistency, $G$ must learn to reconstruct the original image using the generated image $G(x,\tilde{s})$ and the style code $\hat{s}$:

$$V_{cycle} = E_{x,y,\tilde{y},z}[||x - G(G(x,\tilde{s}),\hat{s})||_1], \tag{3.39}$$

Combining those losses, we have the full training objective of StarGAN v2:

$$\min_{G,F,E} \max_{D} V_{adv} + \lambda_{style}V_{style} - \lambda_{diverse}V_{diverse} + \lambda_{cycle}V_{cycle}, \tag{3.40}$$

in which, $\lambda_{style}$, $\lambda_{diverse}$ and $\lambda_{cycle}$ are the hyperparameters that can be used to balance the weight of each term in the total objective.

## 3.7 Disentangled Representation Learning

Learning the features of images in an unsupervised fashion has received attention from the computer vision community for years.

Most methods in the early stage were based on restricted Boltzmann machines [67] and stacked autoencoders [171]. Models in [130] and [106] were proposed for semi-supervised learning and achieved promising results on the MNIST dataset. In [127], a GANs-based method was shown to represent the dataset in a code space where basic linear structures are supported.

Another branch of research uses labeled data to learn disentangled representation. The representation is divided into two parts: one for the given labels and one for other features. Similar fashions of training can be found in different model structures such as bilinear models [161], multi-view perceptron [189], variational autoencoders (VAEs) [88] and adversarial autoencoder [109].

For minimizing the dependency on variation labels, weakly supervised methods were developed. Reed et al. [8] propose correspondence-based training strategies for a higher-order Boltzmann machine consisting of hidden unit groups, and each group represents a factor of variation. A similar technique is applied to VAE [92] to manipulate brightness and pose in images of 3D objects. These two methods share one drawback: they require grouped data points that are difficult to collect in real-life applications.

There are not many works on completely unsupervised disentangled representation learning. In [35], hossRBM is introduced as a generalized version of spike-and-slab restricted Boltzmann machine, which entangles variation factors using its higher-order interactions on latent variables. However, the method is not effective in terms of computation cost.

In InfoGAN [26], Chen et al. develop an extension of GAN, which learns interpretable and meaningful representations of the given dataset in a completely unsupervised manner. The generator of InfoGAN is expressed as $G(z,c)$, in which $z$ is the noise just like in the standard

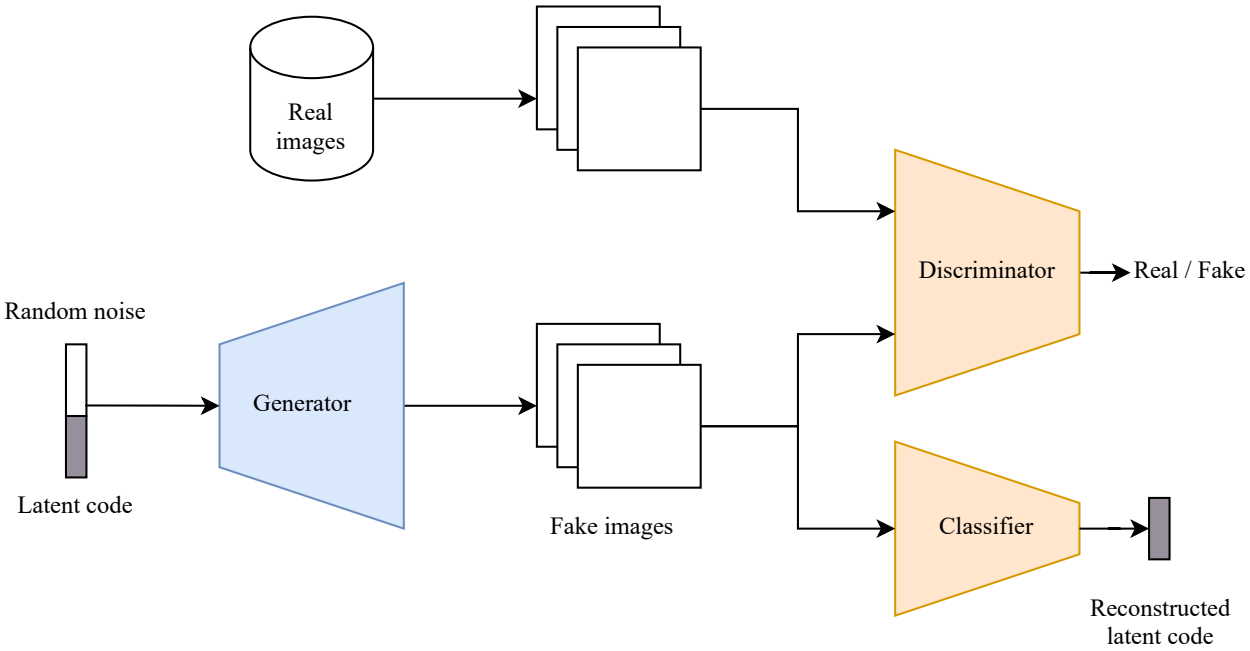
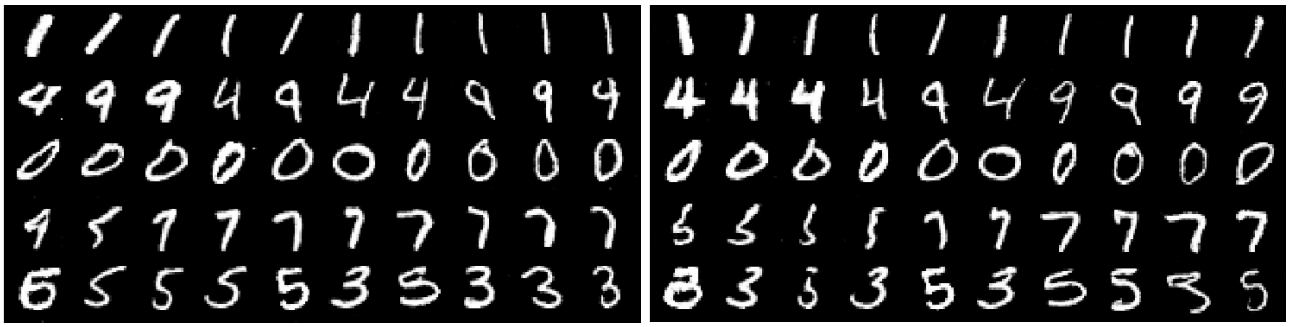



Figure 3.18: InfoGAN structure.

GANs, and $c$ is the latent code. Let $I(c; G(z, c))$ be the mutation information between the latent code and the generator distribution. To make $z$ to store the semantic features of the data distribution, the network must learn to maximize $I(c; G(z, c))$. In fact, it is not simple to directly maximize this mutual information without accessing the posterior probability $P(c|x)$. For solving this problem, $Q$ is proposed as the auxiliary distribution approximating $P(c|x)$, and $L_I(G, Q)$ becomes the variational lower bound of the mutual information between the latent code $c$ and the observations. Combining $L_I(G, Q)$ with the standard GAN loss, we have a mini-max game with a variational regularization of mutual information, which is the objective function of InfoGAN:

$$\min_{G,Q} \max_{D} V_{InfoGAN}(D, G, Q) = V(D, G) - \lambda L_I(G, Q) \tag{3.41}$$

where $\lambda$ is a hyperparameter controlling the importance of the two objectives. Practically, $Q$ is represented in two parts: the categorical latent code in the form of a softmax non-linearity and a continuous latent code in the form of factored Gaussian. The overall architecture of InfoGAN is represented in Figure 3.18.



(a) (b)

Figure 3.19: Manipulating image features using latent codes on MNIST dataset: From top to bottom, the first latent code of the input of InfoGAN varies which changes the digit number. From left to right, the second (a) and third (b) latent code vary which respectively changes the pose and the boldness of the digits.

Experimental results (Figure 3.19) show that InfoGAN successfully learns disentangled and

interpretable representations on multiple datasets without any form of supervision.

## 3.8 Few-shot Domain Adaptation

In the last decay, deep learning has become the favored approach for most image classification tasks given many labeled samples. In reality, however, even though the amount of data is large, labeled data is usually in short. Few-shot learning is a group of techniques that focus on training learning models using a small amount of annotated data. One well-known approach in this group is meta-learning (also called "learning to learn"), where we develop models that can adapt rapidly with a new dataset using a few samples to accomplish certain tasks. Meta-learning can be seen in three main branches: model-based, metric-based and optimization-based. Methods such as [140] and [118] aim to address this problem by developing specific model architectures that make use of the meta-information of images, so they are called model-based methods. Metric-based methods, on the other hand, learn an efficient distance metric that learns to cluster similar data samples into groups [151], [156]. The third type of meta-learning, optimization-based [42], [123], optimizes the initialization of parameters so that the model converges quickly in training. These works show the potentials of meta-learning in few-shot image classification. Still, most of them only work on a single target domain without considering that there are similar datasets that do not share the same distribution but can contain similar features to be exploited.

Domain adaptation methods aim to exploit features on a dataset where many annotated samples are accessible and to test on a target dataset that does not have the same but a similar distribution to the source. We can implement domain adaptation in an unsupervised fashion. In [14] and [71], reweighing and sample-selection techniques are applied to achieve the domain adaptation. In contrast, other works focus on learning the mapping that transforms samples from the source domain to ones in the target domain. These methods do not require labels from the target domain.

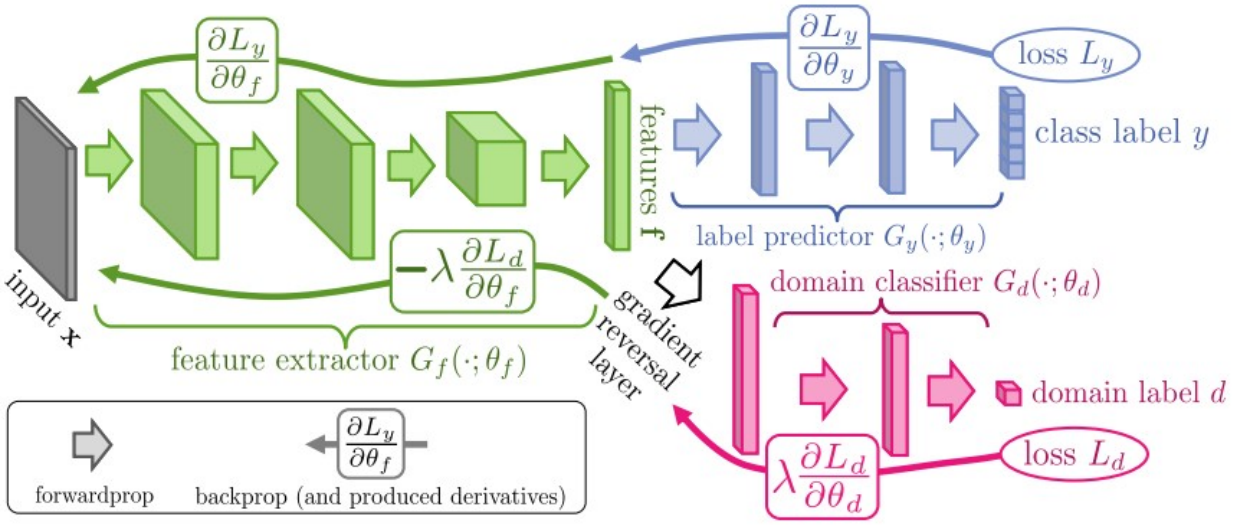



Figure 3.20: DANN architecture consists of a feature extractor (in green), a label predictor (in blue) and a domain classifier (in red). Figure from [49].

DANN [49] aims to match feature space between two distributions by modifying the feature representation. As shown in Figure 3.20, the architecture of DANN has a dual-branches structure. The input image $x$ is given to the feature extractor $G_f$ to be encoded into a vector $f$. Then, the label predictor $G_y$ takes $f$ and predicts the label of the input images $x$ while the domain classifier $G_d$ is trained to predict the domain label of $x$. $G_f$ and $G_d$ are connected

via a gradient reversal layer component, which acts as an identity transformation during the forward propagation while multiplies the gradient by $-1$ during the backpropagation. Thanks to $G_d$, the feature extractor $G_f$ learns to map the images from the two domains into one single distribution, and thanks to $G_y$, it learns to encode the classification features at the same time. After training, the feature extractor and the label predictor are kept as a feed-forward model, which takes an image as input and produces a prediction vector. The method is tested on multiple domain adaptation tasks and achieves promising results.

Labeling a few samples from the target domain and using them for training is not a costly approach but can bring great benefit. Methods following this direction are called few-shot domain adaptation, which aims to train learning models on a source domain (where many annotated samples are available) and test it on a target domain (where just a few samples are labeled). The scenario is prevalent in the real world, but few studies are made in this direction.

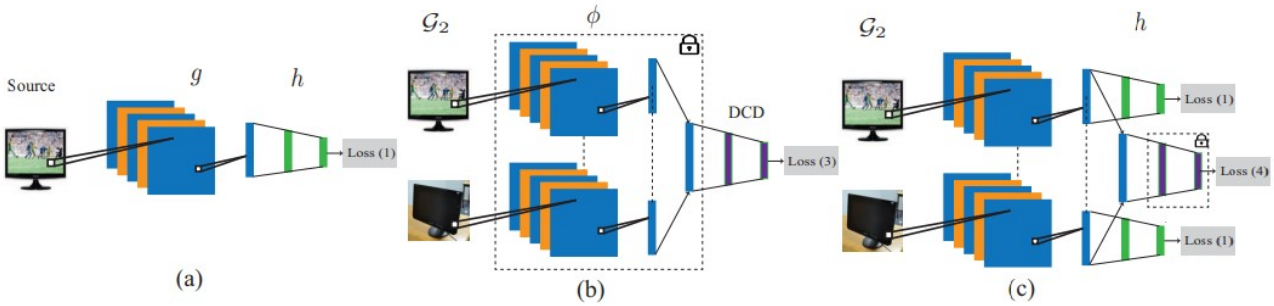



Figure 3.21: The training of FADA in three stages. (a) Features extractor $g$ and label predictor $h$ are trained on the source domain $D_s$. (b) A domain-class discriminator (DCD) is trained while $g$ is frozen. (c) DCD is frozen while $g$ and $h$ are trained. Figure from [117]

Few-shot adversarial domain adaptation (FADA) [117] is a CNN-based that performs a few-shot classification by minimizing both a classification loss and an adversarial loss at the same time. Figure 3.21 explains the training steps of FADA. First, we train a network $f$, which consists of a feature extractor $g$ and a label predictor $h$, using samples from the source domain with the classification loss. Second, we plug $g$ into a domain-class discriminator (DCD) and train this network while freezing $g$. The feature extractor $g$ encodes features of images from the two domains while the DCD is trained with an adversarial loss to realize the domain where the features come from. Afterwards, the DCD is frozen while we train $g$ and $h$ to classify samples from both domains. The method produces promising results even when only a few samples of the target domain are available.

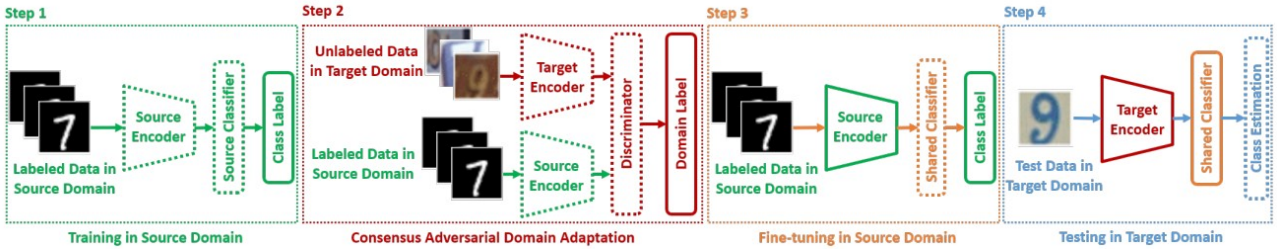



Figure 3.22: Summary training of F-CADA in four stages. Figure from [190]

F-CADA [190] has a similar but more complicated structure. It contains two encoders (feature extractors), two label classifiers and a discriminator for the extracted features. Figure 3.22 illustrates the training process of F-CADA in four stages. Firstly, a source encoder and a source classifier are trained using labeled samples from the source domain. They should be well-trained as the number of labeled samples in this domain is large. At the second stage, a target encoder is trained to transform target data to a feature space, while the source encoder is also fine-tuned to map the source data to this space. A domain discriminator is trained at

the same time to predict the target of the feature while the two encoders are trained to fool the network. This step ensures features extracted from the two domains are represented similarly in the feature space. For the third stage, a classifier takes outputs of the source encoder and learns to predict the image label from the source domain. This classifier is supposed to perform well on the outputs of the source encoder thanks to a large number of labeled samples. Finally, the classifier is plugged into the target encoder's output layer to classify testing target samples. The advantage of F-CADA over FADA is that it has two independent encoders for encoding images of two domains. Still, they both map the images into the same feature space. The method holds state-of-the-art performance on multiple few-shot domain adaptation tasks.

## 3.9 Deep Learning in Dentistry

The continuous development of artificial intelligence and the boom in dental diagnostic technology in recent years have opened up many opportunities for the application of artificial intelligence in this field, especially where digital images are available. Photographic and radiographic images are both used in the domain of dentistry, but radiographic images are used more often because they also reveal the hidden parts of teeth, which are important for diagnosis. Using radiographic images, CNNs are trained for multiple computer vision tasks such as classification, recognition and segmentation of anatomical structures or dental complications. For example, deep learning models are used for recognizing teeth in cone-beam computed tomography (CBCT) images [114], periapical films [24] and panoramic radiographs [163]. In [68], a deep learning model is proposed for assessment of molar root morphology on panoramic radiographic images. CNNs are also used to define anatomical structures before planning dental surgeries. In [170], a deep learning model is trained for segmenting the inferior alveolar nerve (IAN) and the roots of third molars on dental panoramic radiographs for the purpose of planning before wisdom tooth extraction.

Researchers also apply CNNs for detecting dental caries, inflammation, and periodontal diseases on multiple types of digital images. Near-infrared trans-illumination (NITI) imaging is getting attention as an effective method for capturing and detecting dental caries in their early phase. In [20] and [142], neural networks are trained to automatically detect and localize dental lesions in this type of image with an accuracy higher than 80%. Radiographs are usually used in addition to visual observation to identify the deformation of teeth when enamel and dentin are damages. In [98], CNNs are shown to detect and diagnose dental caries in panoramic dental radiographs effectively. In another research [91], CNNs are applied to detect periodontal bone loss (PBL) on radiographic images, which is an important element for evaluating dental health. Oral cancer is a serious problem, but it can be effectively cured if it is detected and treated in time. CNNs are also used for detecting cancerous and precancerous lesions. Especially, in [166], a low-cost and portable solution is proposed for practicing oral cancer screening with the help of a smartphone camera. This is one of a few applications CNNs in the domain of dentistry being done outside of the laboratory environment. In [4] deep learning, object detection models are trained to detect radiolucent lesions of the mandible in panoramic radiographic images and achieves a high sensitivity (0.88).

These studies have shown the huge potential of deep learning for processing and diagnosing images in the domain of dentistry. However, most of them share some basic drawbacks. Firstly, these deep learning models are trained using small datasets (hundreds of samples), leading to a high risk of overfitting. Lack of diversity in the training data makes it impossible to extend these works widely. Secondly, up to 90% accuracy can be considered high in many conventional applications but is quite low in the medical field in general and in dentistry in particular. Certain errors in the diagnostic process can also put a patient's health and life in danger. Thirdly, the majority of CNN's applications in the dental field mentioned above are limited to an experimental environment and are not incorporated into an applicable pipeline as a complete product or service.

# Chapter 4

# Detecting Gingivitis in Oral Images Captured by Smartphone Cameras using CNN

## 4.1 Introduction

Periodontal diseases (PDs) are inflammatory diseases affecting the gingiva and the supporting tissues of teeth and also the primary cause of tooth loss in adults. Research shows that the chance of having PDs increases when a person is under orthodontic treatment [172]. If not treated promptly, PDs can lead to serious diseases such as diabetes, pneumonia due to inhalation, strokes and cardiovascular disease [128]. In contrast, the early stage of PDs, often known as gingivitis, can be treated much more easily than when it has progressed to severe levels. Therefore, orthodontic patients need to be frequently diagnosed to detect the early signs of PDs - commonly known as gingivitis - to provide timely dental care. Thanks to the rapid increase of worldwide smartphone ownership, remote health care applications have become more accessible than ever. With smartphones, patients can take pictures of a body part and send them to doctors or even computer programs for diagnosis [41].

In this work, we detect gingivitis from images captured by phone cameras using convolutional neural networks (CNNs). Experimental results show that our classifier can distinguish between healthy and inflamed gingivae with an accuracy of 90.88%, and the area under the ROC curve (AUC) is 95.52%.

## 4.2 Dataset

Images used in this work were collected from the Dental Mind database. We are using 592 records from 280 patients for this research, acquired in six months in 2018. Each record consists of eight images captured by a smartphone from different poses: left, front and right for opened mouth; left, slightly-left, front, slightly-right and right for the closed mouth, with the help of a cheek-retractor (Figure 4.1). Photos in our dataset were taken by various types of phones in different lighting conditions. Note that our work has been done locally at Dental Mind, and patient information was removed from the dataset. Dentists annotate the dataset at the level of teeth, which means if gingivitis appears on the region of the gingiva surrounding a tooth, we mark the tooth as gingivitis. Otherwise, the tooth is marked as healthy. In the processing phase, we crop a pair only if both teeth have the same label. Gingivitis appears in 274 records in the dataset, while the other records are completely healthy. We obtain 19978 patches from all records after preprocessing.

## 4.3 Methodology

This section describes a pipeline including a preprocessing phase being composed of masking and cropping, then a deep neural network to predict the existence of gingivitis in a record.

### 4.3.1 Preprocessing

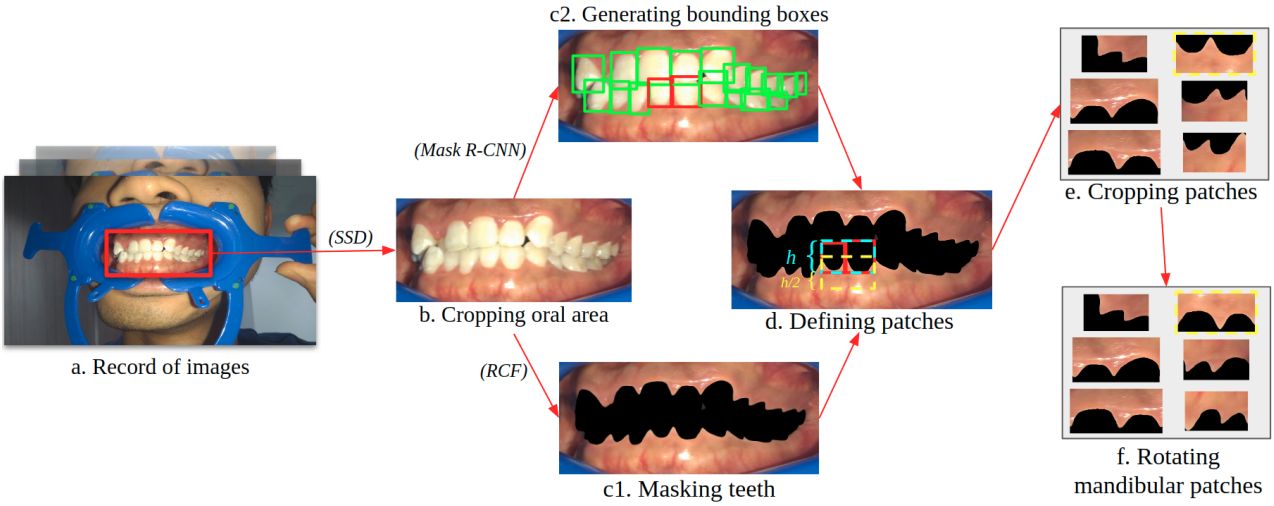



Figure 4.1: Preprocessing procedure

We aim to train a CNN from patches containing gum region between pairs of teeth, where gingivitis usually appears. We develop a preprocessing procedure to extract those patches from the dataset, consisting of hundreds of records. Each record contains oral images of a patient from eight poses (Figure 4.1a). Images resolutions vary from 611 × 328 up to 3774 × 2261 pixels as they are captured by patients using different devices.

Images from records (Figure 4.1a) are cropped to the region of interest (ROI) containing gums and teeth (Figure 4.1.b). The ROI is defined by Single Shot MultiBox Detector (SSD) [47], a popular object detection algorithm that we fit and use to detect the oral region in images. Since textures of teeth do not contribute to gingivitis detection and might cause the network to learn biases such as braces (because gingivitis often occurs during orthodontic treatments), we mask out the region of teeth in the image using a contour detection and segmentation method, namely Richer Convolutional Features (RCF) [104] (Figure 4.1.c1). Next, Mask R-CNN [63], a deep learning segmentation technique applied for teeth detection to obtain bounding boxes of all teeth (Figure 4.1.c2). We ignore the bounding boxes of molar teeth because they are not usually well-captured.

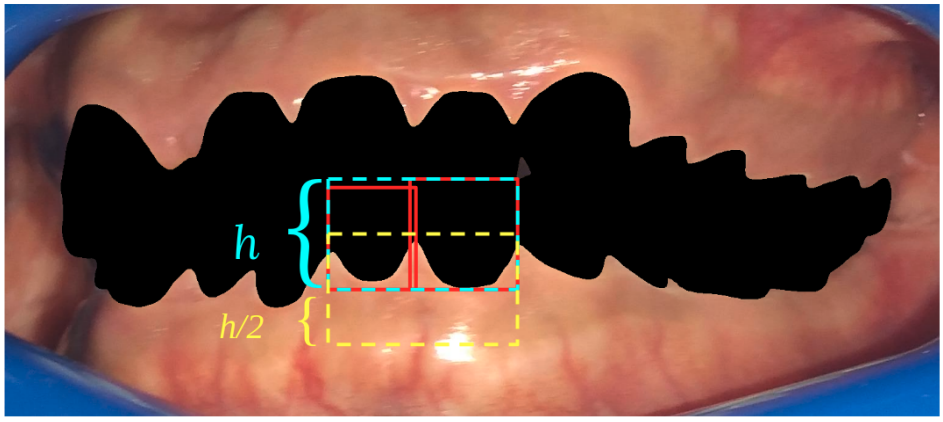



Figure 4.2: Cropping pairs of teeth

As gingivitis is more visible at the gum tissues between pairs of teeth, we crop patches of images containing the gum region below and between two teeth. As illustrated in Figure 4.1.d and Figure 4.2, for each pair of bounding boxes (red), we compute a larger box (cyan) that covers the two boxes. Then, the box is shifted toward the gum (upward if the pair of teeth comes from maxillae, and downward if it is from mandibular). The border of the final patch is in yellow. The shifting distance equals half of the height $h$ of the large box. Finally, we rotate all mandibular patches by 180 degrees (Figure 4.1.f) and resize all patches to $128 \times 128$ (RGB). To avoid deformation, we apply zero-padding to all patches to make them square-shaped before resizing.

### 4.3.2 Classification model

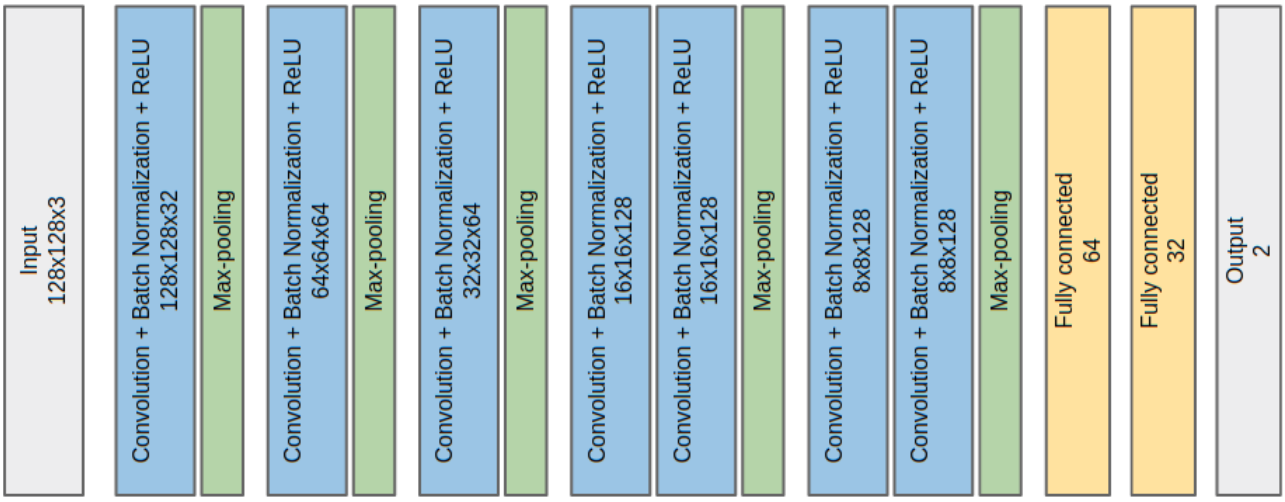



Figure 4.3: Gingivitis classifier architecture

We train a CNN which classifies those $128 \times 128$ patches in two classes: *healthy* and *gingivitis*. The network architecture consists of multiple blocks: Convolution + Batch normalization + ReLU, Max-pooling following and Fully connected layers at the end as shown in Figure 4.3. Two neurons at the end of the network represent the classification probability of the input image on two classes: *healthy* and *gingivitis*. The classifier's input will be predicted to the class, which is represented by the output neuron carrying a larger value. The sum of two neurons is 1.0, so if one of the two output neurons has a value greater than 0.5, the input patch is predicted to the class that the neuron represents. Since each pair of teeth is captured from several views, we feed all views of the pair through the CNN and obtain a prediction for each view. The final prediction of a pair of teeth is computed as:

$$p_{final} = \frac{1}{N} \sum_{i=1}^{N} p_i \tag{4.1}$$

where $p_i$ denotes the prediction result of a single view, and $N$ is the number of views the pair appears.

### 4.3.3 Data augmentation using style-transfer GAN

Style-transfer networks have been used recently to deal with the problem of lacking data in medical image analysis [147]. In this work, we use a style-transfer model named MUNIT [73] in order to generate *gingivitis* patches from *healthy* patches and vice versa. MUNIT consists of an encoder $E_i$ and a decoder $G_i$ for each data domain $X_i (i = H, G)$ ($H$ represents *healthy* class and $G$ represents *gingivitis* class). The encoder $E_i$ transforms input $x_i$ to become content code $c_i$ and style code $s_i$ as following:

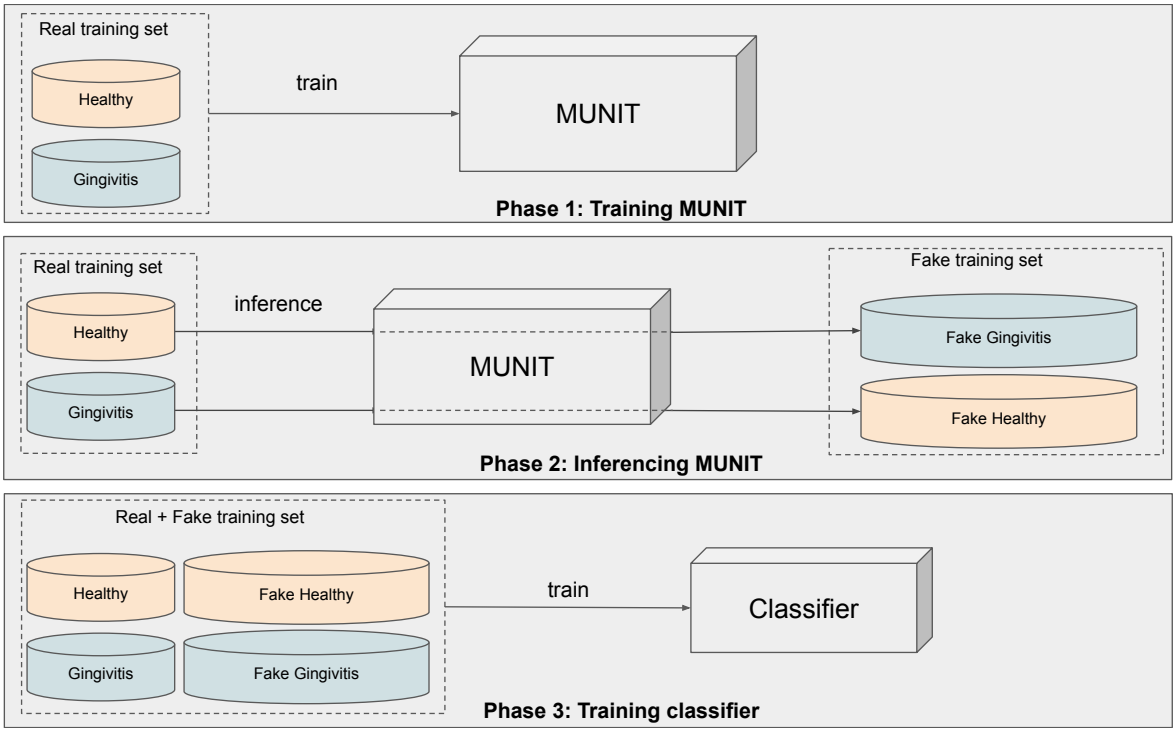



Figure 4.4: Using MUNIT for augmenting training data to train the gingivitis classifier

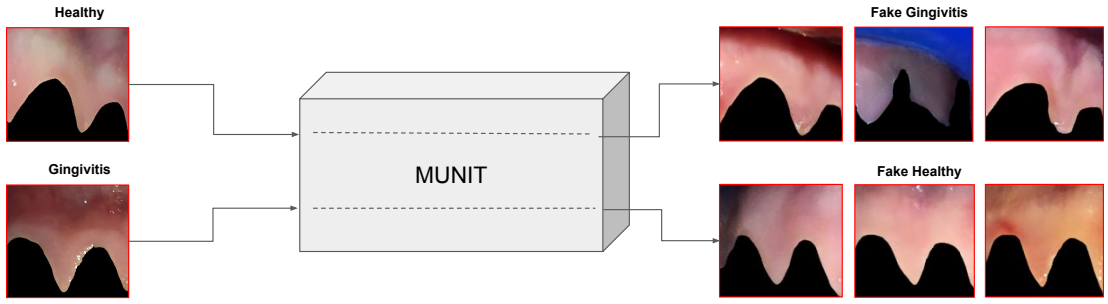



Figure 4.5: Some examples of fake samples generated using MUNIT

$$(c_i, s_i) = (E_i^c(x_i), E_i^s(x_i)) = E_i(x_i). \tag{4.2}$$

We implement style transformation by swapping the encoder-decoder pairs. For instance, given a *healthy* patch $x_H$, we can synthesize an artificial *gingivitis* that is denoted as:

$$x_{H \to G} = G_G(E_H^c(x_H), s_r) \tag{4.3}$$

in which, $s_r$ is a style latent code drawn from prior distribution $q(s_r) \sim N(0, I)$. Similarly, artificial *healthy* patches can be synthesized as

$$x_{G \to H} = G_H(E_G^c(x_G), s_r) \tag{4.4}$$

where $x_G$ is a patch from domain *gingivitis*. By modifying $s_r$, we can receive multiple versions of the output which share the same content but differ in style.

We apply MUNIT for data augmentation as described on the left of Figure 4.4. First, we train MUNIT with training data of two classes. Then, we apply the pre-trained MUNIT for synthesizing artificial data. From each real patch as input, we synthesize three outputs (Figure 4.5) by using three different values of $s_r$. Finally, we combine the original data with artificial data to train our classifier.

Table 4.1: Accuracy (%) and AUC (%) of classifiers trained with original data and on enlarged data with two prediction techniques: single view and multiple views.

| | Single view | | Multiple views | |
|---|---|---|---|---|
| | Accuracy | AUC | Accuracy | AUC |
| Real data | 88.81 ± 3.55 | 94.98 ± 2.82 | **90.88 ± 3.25** | **95.52 ± 1.88** |
| Real + Artificial data | 88.15 ± 4.07 | 94.81 ± 2.91 | 90.43 ± 3.64 | 95.51 ± 2.00 |

## 4.4 Experiments

### 4.4.1 Training classifier

Our classifier is trained with a batch size of 128 and Adam optimizer [87] with a learning rate of $10^{-4}$ on 3500 epochs. We validate the model every 15 epochs and compute the average result at the end of the training. We construct two versions of the classifier: one is trained on the given training set, and the second one is trained on the same training set mixed with artificial images generated by MUNIT. Using this set of hyperparameters, it takes 50 hours to finish the training on a GeForce GTX 1080 Ti graphic card.

### 4.4.2 Training MUNIT

The multimodal style-transfer network is trained using the same hyperparameters as described in the original paper [73] except for the number of epochs and batch size. We train the model for 2800 epochs with a batch size of 14 instead of 1 million iterations and batch size of 1 for all datasets as in the paper. We use larger batch sizes and fewer training steps for optimizing the computing resource. Input/output size of the model is also reduced from $256 \times 256$ to $128 \times 128$. MUNIT and the classifier share the same training set (Fig. 4). We use the same device that trained our classifier to train MUNIT, and it took 9 days for the training.

### 4.4.3 Cross-validation

The accuracy and area under the ROC curve (AUC) of the classifiers are estimated by 10-fold cross-validation. Note that MUNIT learning is included in the cross-validation loop to avoid that the artificial images overfit the test set. Data is divided at the level of records to assure that images coming from one record will never appear in the training set and validation set at the same time. The data of each fold contains about 18,000 patches and 2,000 patches for training and validation, respectively.

## 4.5 Results and Discussion

The classifier is evaluated using a cross-validation procedure by accuracy and area under the ROC curve (AUC). As shown in Table 4.1 and Figure 4.6, the classifier achieves better accuracy and AUC by combining the predictions of multiple views. This phenomenon is similar to reality, where doctors also need to look at patient teeth from several angles for a proper diagnosis.

The table also shows that adding artificial training data using the style-transfer network does not improve classification performance in this case, even though they look convincing on the right side of Figure 4.5.

We also see that the model performs differently based on the location from where patches were cropped. The model achieved an AUC of $96.17 \pm 2.23$ when evaluated with patches among incisors and canines and an AUC of $94.25 \pm 2.25$ when evaluated with patches among canines and premolars. Since gum regions from the front usually look clearer than ones from the side and the back, they are easier to be recognized by the classifier. This result is reasonable because teeth from the front are displayed more clearly and frequently.

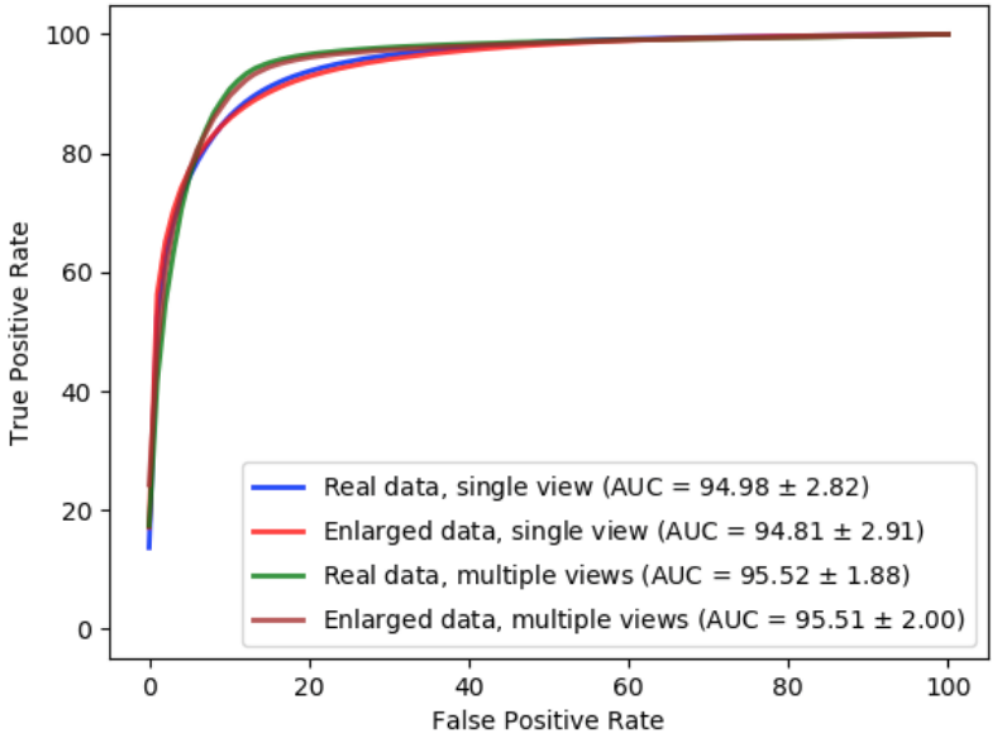



Figure 4.6: ROC curves and AUC of the classifier when being trained on real data versus enlarged data, evaluated using single view and multiple views.

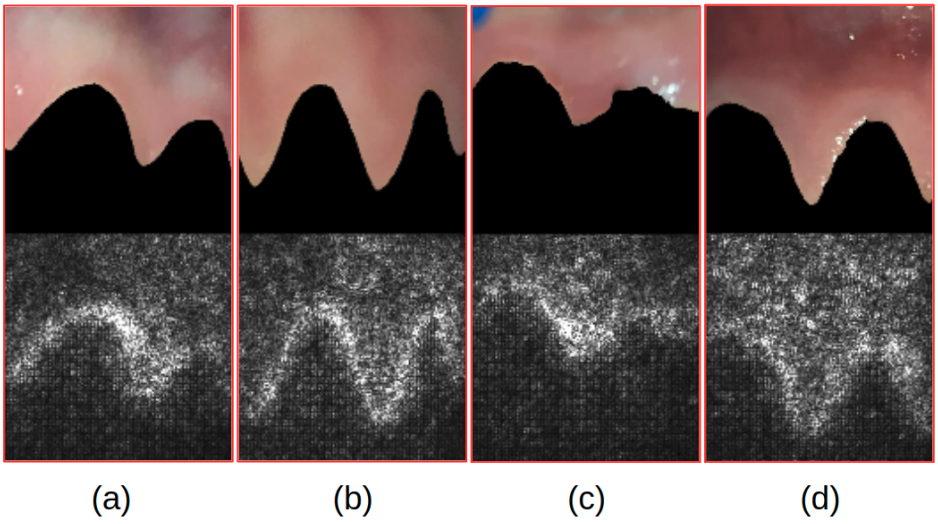



Figure 4.7: Examples of saliency maps of healthy samples (a) & (b) and gingivitis samples (c) & (d). The brightness of pixels represent how much they contribute to the prediction.

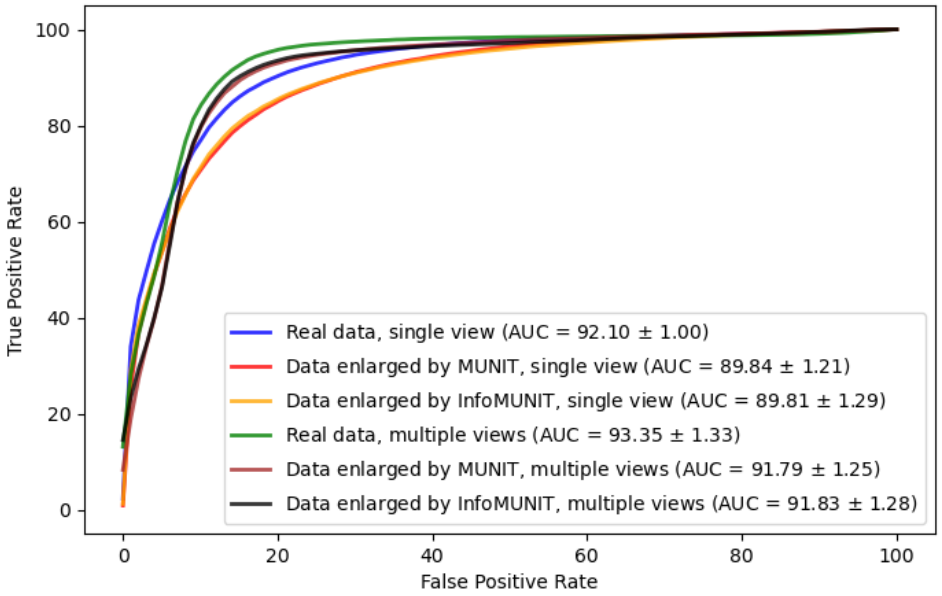



Figure 4.8: ROC curves and AUC of the gingivitis classifier when being trained on only real data, enlarged data of MUNIT and enlarged data by InfoMUNIT. The result is evaluated in two modes: single view and multiple views.

### 4.5.1 Saliency Map

Recent work [51] showed that CNNs trained on the ImageNet dataset tend to be biassed towards texture rather than object shapes as we usually think.

Even if we obtained a high accuracy of gingivitis prediction, we have to check if our classifier has really learned the concept of gingivitis or based its decision on the artifact in the training set. A common approach to interpreting the decision of a CNN is to visualize the saliency maps. Saliency maps compute the gradient of the output prediction with respect to the input image and highlight image pixels that contribute the most to the predicted output [149].

As can be seen in Figure 4.7, there is a high distribution of white dots on the boundary between gum and (masked) teeth. The density of dots on the gum region is also high but not as much at the boundary. This result suggests that both shape and texture contribute to the classification result, but the shape of the border between teeth and gum seems to be the most important component. The cause of this result may come from different lighting conditions between photos in the dataset. The color of the gums can be displayed more boldly in one camera but lighter in other cameras. In contrast, the shape of the gums does not change even if images are taken by different devices and under various conditions. This is coherent to the conclusion of an orthodontist in our team, stating that the most recognizable sign of gingivitis is swollen gum.

### 4.5.2 Data augmentation using InfoMUNIT

Later, we develop InfoMUNIT as an extension of MUNIT for disentangled multimodal image-to-image translation. The method successfully generates more realistic and diverse outputs than MUNIT. Therefore, applying InfoMUNIT for data augmentation would be an interesting experiment. For the limitation of time, we do not apply the cross-validation procedure for this test, so we just randomly use one of the prepared fold of data. As can be seen in Figure 4.8, InfoMUNIT does not make a lot of differences compared to MUNIT. They both do not improve the classification performance but, in fact, reduce it, suggesting that this approach is not quite suitable for data augmentation.

## 4.6 Conclusion

In conclusion, this work proposes a pipeline to detect gingivitis from photos taken with smartphone cameras with high accuracy. Our approach opens new ways for the diagnosis and follow-up of PDs. We can imagine the following use-case : Initially, a patient wears a cheek-retractor and takes images of his/her mouth from different angles. Then, images are sent to a clinic to be automatically preprocessed and given to the classifier to detect gingivitis. Finally, the results are sent back to the patient. Moreover, if gingivitis is detected in the images, the system sends the images and results to the dentist and triggers an appointment if needed.

For future work, some questions will be addressed. We will investigate why enlarging the training dataset with a style-transfer network does not improve the classification performance, although the artificial images look very similar to real images. We will also improve the model performance by adding preprocessing steps to emphasize the images' structural information using techniques such as adjusting histogram or edge detection.

# Chapter 5

# Adversarial Domain Transfer for Improving Dental Crowding Detection

## 5.1 Introduction

Crowding of teeth is a dental problem that happens when there is not enough room in the mouth for all teeth. It makes teeth bunch up, overlap or even twist. Patients who have crowded teeth usually have difficulties in oral hygiene, giving the harmful bacterial the opportunity to grow and lead to gum disease and tooth decay [65].

At Dental Mind, we develop a solution for remotely tracking the orthodontic treatment progress through a mobile application. It would be helpful to train a neural network that automatically detects and estimates the level of teeth crowding by looking at oral images captured by this application.

The regular problem that we face in this project is lacking annotated data. In order to train a network to predict the existence or the level of dental crowding, we need a data set containing pairs of one photo and one ground-truth label. Labeling this type of data is time-consuming and costly, so we do not have many samples like that. The label can be estimated by looking at the taken photos but cannot be highly accurate without access to the position of teeth. An alternative solution is to implement the measurement in 3d-models of teeth, but in this case, labels are paired with a 3d-model instead of a colored image. However, a CNN is trained on this type of data will not work well on colored images. This issue can be considered as a domain adaptation problem. In this project, we aim to use a generative network called Pix2pixHD [173], to transform projections of 3d-models of teeth into real-looking images, which can be used to train a deep classifier that learns to detect the existence of teeth crowding in real images.

Our goal is to learn to detect the existence of dental crowding from oral images. The work can be extended in the future to estimate the levels of crowding by using more output categories or regression labels.

## 5.2 Methodology

This section describes a deep classifier that is trained to detect dental crowding and a GAN-based model that synthesizes additional training data from 3d-models.

### 5.2.1 Classification model

We train a deep CNN classifier to distinguish between crowded samples and no-crowding samples. As the main purpose of this work is to study the effectiveness of using synthetic data, we choose a simple architecture for the classifier so that it will not take too long for training and tuning hyperparameters. The architecture of the classifier is visualized in Figure 5.1. It

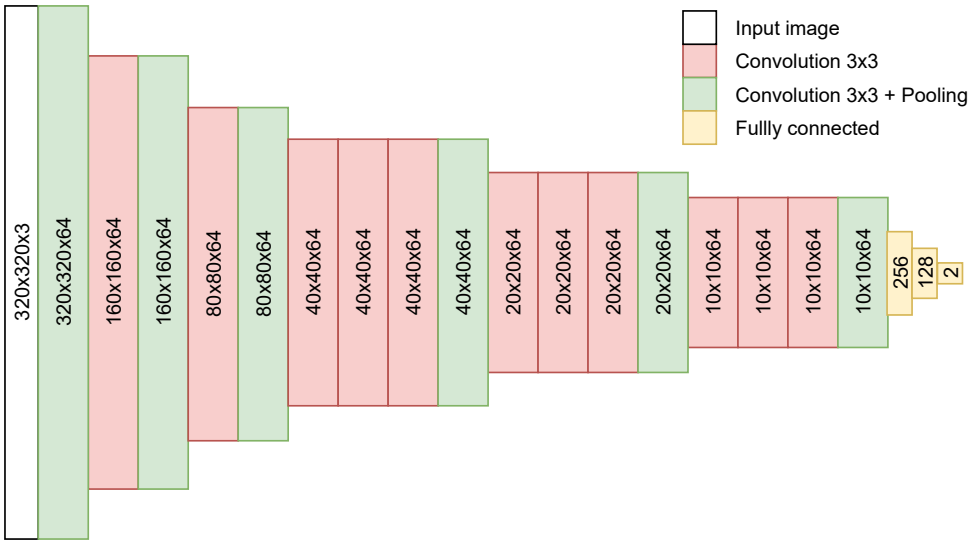



Figure 5.1: Crowded teeth classifier architecture.

has a simple architecture consisting of convolutional layers and max-pooling for downsampling. All convolutions are followed by batch normalization and ReLU activation. Compared to the gingivitis classifier in the previous chapter, this model is much deeper for the much larger size of the input images.

### 5.2.2 Domain adaptation for data augmentation

To deal with the lack of training data, we take photos of 3d-models and then use Pix2pixHD [173] to generate realistic details on those gray-scaled images. Those generated images are used as additional training data for the classifier. Our pipeline for enlarging the training dataset is described in the session of Experiments.

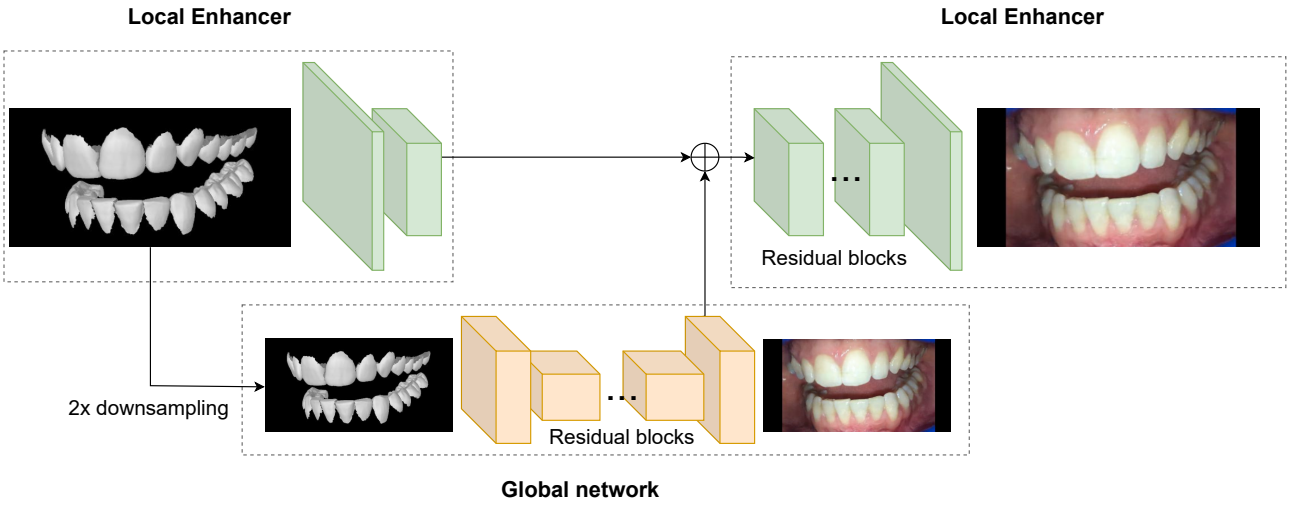



Figure 5.2: Coarse-to-fine generator. The residual network being trained in the first phase is in yellow. Layers which are appended in the second phase are in green.

As described in Section 3.6.1, Pix2pixHD is developed based on Pix2pix [80] to translate images on a higher resolution. Compared to Pix2pix, the model has significant improvements. Firstly, the coarse-to-fine generator of Pix2pixHD is trained in multiple phases. In the first phase, a residual network is trained on low-resolution images to generate low-resolution images. In the second phase, residual layers are appended to the trained network to be jointly trained. The second phase can be repeated to produce higher resolution images. Figure 5.2 visualizes

this process. Secondly, it uses discriminators that have an identical network structure but operate at different image scales. For our case, we train two discriminators: one for the global network's output and one for the local enhancer's output. Thirdly, Pix2pixHD learns to extract features from different layers of its discriminator and match these features from the real and generated image, not only from the output layer as in Pix2pix.

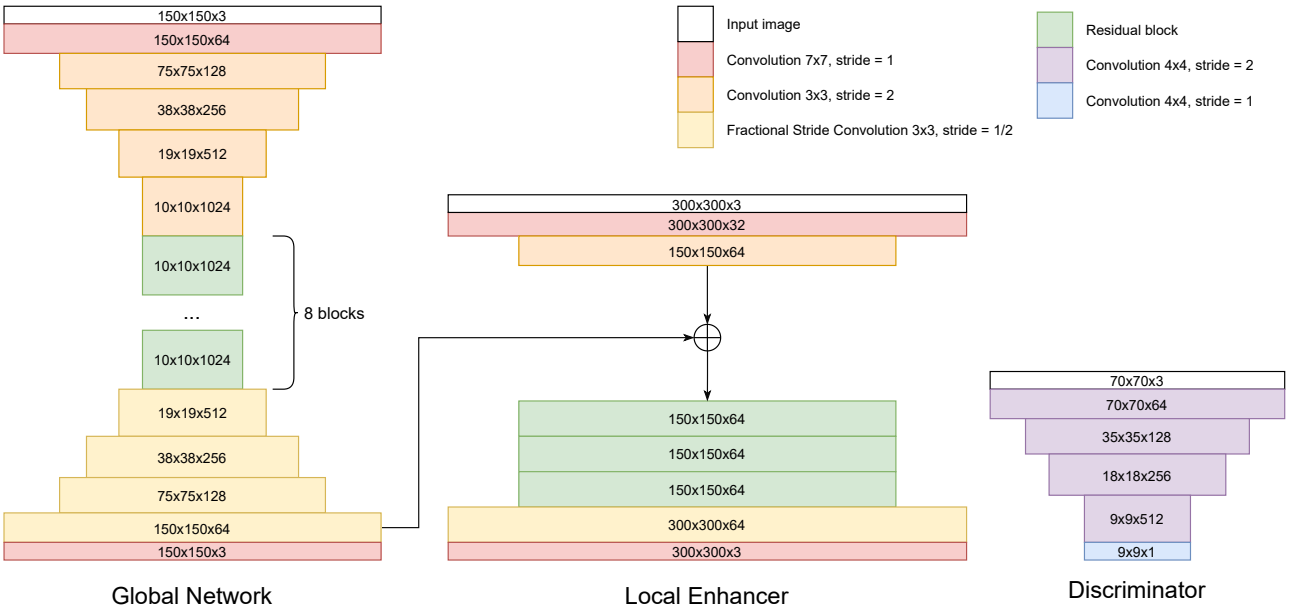



Figure 5.3: The architecture in detail of each sub-network in Pix2pixHD: Generator including Global Network (left) and Local Enhancer (middle), and Discriminator (right). For our experiment, the input image is in RGB with the size of 300×300.

The detailed architecture of each sub-network in Pix2pixHD can be seen in Figure 5.3. The generator of the network is inspired by the one from CycleGAN [188] but divided into a global network and a local enhancer. The global network is basically similar to CycleGAN, with convolutional layers with stride for downsampling, residual blocks, fractional stride convolutional layers for upsampling and another convolutional layer at the end where it produces the low-resolution output image. The convolutional layers are followed by instance normalization and ReLU. Reflection padding is used to avoid artifacts at the image boundary. Note that the size of the global network input is already divided by two for each side. The full-size image is given to the local enhancer, which consists of convolutional layers where it is down-sampled, residual blocks, and fractional stride convolutional layers to restore the original size. The last feature map in the global network is added to the input of the first residual block in the local enhancer. The structure can be easily extended to work with even larger size images by creating another local enhancer that has the input of its first residual block added with the last feature map in the first local enhancer. Instance norm, ReLU and reflection padding are also used for convolutions of the local enhancer, just like in the global network.

Pix2pixHD adopts the Patch-GAN [80] architecture for its discriminator. We randomly crop $70 \times 70$ patches from real images and generated images and train the discriminator to predict if the pair of input is real or fake (see Figure 3.11). The network simply consists of four $4 \times 4$ convolutional layers with stride $= 2$ and one convolution with stride $= 1$. All those convolutional layers are followed by instance normalization and leaky ReLU as recommended in [80]. The last convolution produces a one-dimensional output.

## 5.3 Experiments

### 5.3.1 Dataset

The dataset for this study is collected from the Dental Mind database. All experiments are implemented at the company to keep the data safe. We use three datasets, namely A, B and C, for our experiments.

Dataset A consists of dental 858 records (425 records are labeled as no-crowding; 433 records as crowding). Each record consists of 6 images taken from 6 views being represented in Fig 5.4. We use 258 records (125 no-crowding records and 133 crowding records) which means 1548 images for training and 600 records (3600 images) for validation. We resize images to the size of $300 \times 300$ (RGB) and apply zero-padding to make them squared shape. Images from different records are mixed and fed to the CNN.

Figure 5.4: Six views selected from a record. First column: left and front-left, second column: front and front-bottom, last column: right and front-right.

Figure 5.5: Six projections of a 3d-model. First column: left and front-left, second column: front and front-bottom, last column: right and front-right.

Dataset B consists of 516 3d-models of teeth (250 models are labeled as no-crowding; 266 models as crowding). We take photos of those 3d-models from the same six views as photos in dataset A were taken (Figure 5.5) to get 3096 images at the end. Those images will be transformed to real-looking images by Pix2pixHD, then mixed with dataset A  train the dental crowding detector. We resize those images to the size of $512 \times 256$ (RGB) and apply zero-padding to make them in the expected shape without scaling them.

Dataset C is collected for training Pix2pixHD and contains 7064 pairs of one 3d-model and one real image. The positions of teeth in 3d match with the positions of teeth in the corresponding photos. We use 6535 pairs for training and the rest for validating the model during training. Images are resized to $512 \times 256$.

### 5.3.2 Training

Pix2pixHD is trained for transforming 3d models of teeth to corresponding realistic images. The model is trained on dataset C, using Adam optimizer [87] with a learning rate of $2 \times 10^{-4}$ for the first 200 epochs. Over the next 200 epochs, the learning rate is linearly decayed to zero. After being trained, the model is used to transform 3096 projection images in dataset B into real-looking images Figure 5.6. Those synthetic images will be used as additional training data for the crowded-teeth detector.

Figure 5.6: Six synthetic samples generated by Pix2pixHD from 3d-models in Figure 5.4. First column: left and front-left, second column: front and front-bottom, last column: right and front-right.

The crowded teeth detector is trained for one million iterations, with a batch size of 16 learning-rate of $10^{-4}$ and Adam optimizer [87]. Validation is done every 2000 training iterations. In our experiments, the classifier is trained multiple times with different training datasets. In the first experiment, we train the classifier with 1548 real samples from dataset A with no synthetic samples. In the second and third training, we respectively add 1548 and 3096 synthetic images (3d-models projections transformed by Pix2pixHD) to the training data.

In order to make use of the 3d-models, we slightly rotate those models by 2 degrees on the x, y and z-axis to generate more images before sending them to Pix2pixHD to be transformed into real-looking images. This action increases the number of synthetic images in the training set by 27 times, leading to an unbalanced ratio between real and synthetic samples. In order to close this gap, we duplicate all real samples seven times and 15 times. The results in the next session show that it is helpful to do so.

### 5.3.3 Results and Discussion

We evaluate our classifier's performance based on its maximum and average classification accuracy through the training process. Figure 5.7 shows that the classification performance is increased when we add more synthetic data to the training set. This result suggests that Pix2pixHD successfully learned to map samples from the source domain (projections of 3d models) to the target domain (realistic images).

We also study the effect of data augmentation in 3d environment by slightly rotating the teeth models across the three axes. Applying augmentation in 3d environment helps to increase the number of synthetic samples, which means increasing the size of training data. However, as can be seen in Figure 5.8, when the amount of synthetic data is too large, the performance of the model drops. For this reason, duplicating the set of real samples is a considerable approach for this problem. The results show that the ratio between the number of real samples and synthetic samples has a significant impact on the classification performance.

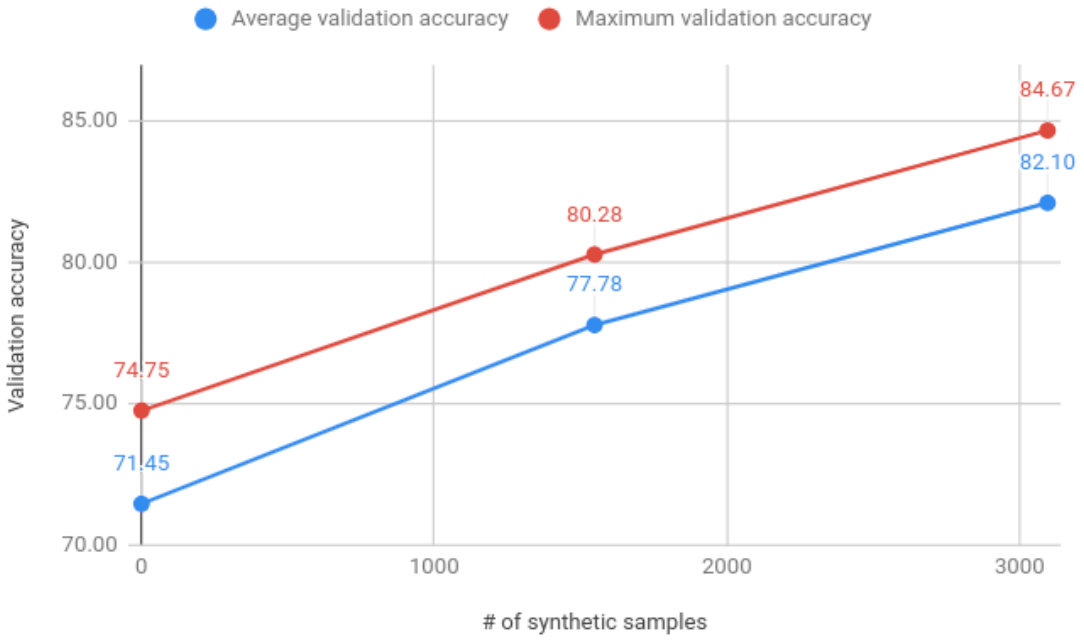



Figure 5.7: Average and maximum classification accuracy over the number of additional synthetic samples.

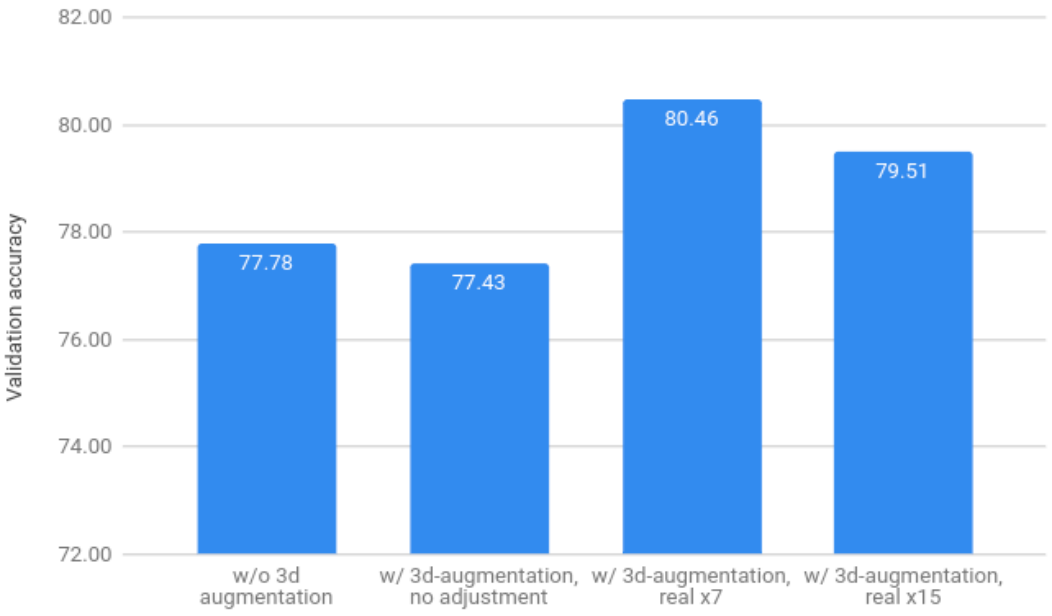



Figure 5.8: The effect of 3d-augmentation on the accuracy of the crowded teeth classification model.

### 5.3.4 Conclusion

In this work, we train a deep neural network that detects the existence of crowding teeth from oral images taken from smartphone cameras. Then, we study the effect of using synthetic data created from projections of 3d-model and image-transformation GANs. Experimental results show that our synthetic data boosts the classification performance of the dental crowding detector. In future works, we will upgrade the classifier to be able to predict the level of the crowding problem and also study the effect of synthetic data in this scenario.

# Chapter 6

# Aligner generation using generative adversarial networks

## 6.1 Introduction

Orthodontic treatment is the process that aims to correct the positions of teeth to improve the facial appearance, correct problems with the bite, and prevent and solve dental complications. In most cases, the patient can choose between multiple types of treatments that are different in multiple factors such as duration, convenience, price, appearance, etc. Normally, after the first examination, including x-ray scans and 3d scans of the teeth, the orthodontist can quickly outline one or more treatment plans using different technologies such as metal braces, ceramic braces, lingual braces, aligners and so on. The orthodontist also gives a comparison explaining the advantages and disadvantages of each method based on multiple factors listed above. Different patients have different preferences. While certain patients are willing to sacrifice time and convenience in return for a lower price, some others tend to pay more to have a discrete and quick treatment. Therefore, it is important for the orthodontist to clearly explain all the characteristics of all the possible options and provide useful advice according to the situation so that the patient makes a decision that they will not regret later when the treatment is already started.

However, in many cases, patients are still hesitant to choose treatment because they are concerned that it is not suitable for their face. For example, ceramic braces look suitable on the face of patient A, but it does not mean that patient B will feel satisfied wearing them, which may increase the chance that the patient will abandon the treatment [120]. Unlike clothes, unfortunately, braces cannot be simply "tried on", so the patient can only "imagine" how they will look with the appliances.

It would be helpful if a computer vision solution can locate teeth in a photo and generate appliances according to their positions. Such application must satisfy two requirements. Firstly, all teeth must be detected and located precisely. Second, the generated details must look realistic, sharp and match the lighting and color conditions of the rest of the photo.

The contributions of this work are following. Firstly, we apply two widely used unpaired image-to-image translation methods for generating details of aligners on images of teeth. Secondly, we study characteristics and compare their performance to find out the most suitable method for the application at Dental Mind. Thirdly, with team R&D of the company, we propose a pipeline with pre/post-processing to apply the generation model for production.

## 6.2 Dataset

We prepare a dataset from 5739 photosets of 1087 patients who have been treated by wearing aligners. About 3 to 5 images are retrieved from each photoset. The number of images from photosets is not the same because low-quality images are rejected. From each photo set, we

get one image without an appliance and one image with aligners. As a result, we collect 23384 images, of which 11692 images are without an appliance, and 11692 images contain teeth wearing aligners. For each category, 11192 images are used for training and 500 images are used for validation.

The dataset contains images taken with the ScanBox and without the ScanBox because they look alike after being cropped. They are cropped and selected as described in Section 2.4.1. We use only images of the front and slightly turn views. The size of images varies, but most of them are of high quality (about 2000 pixels of width and 1000 pixels of height).

## 6.3 Methodology

In this section, we firstly describe the overall system of the application. Then we explain the architecture and the training of the two image translation methods used to generate aligners on dental images.

### 6.3.1 Pipeline

in Section 2.4.3, we explain the idea of the Appliance Selection application by Dental Mind company which aims to generate multiple types of orthodontic appliances on given portrait photos of a patient. It supports metal braces, ceramic braces and transparent aligners. In the scope of this work, we focus on generating transparent aligners.

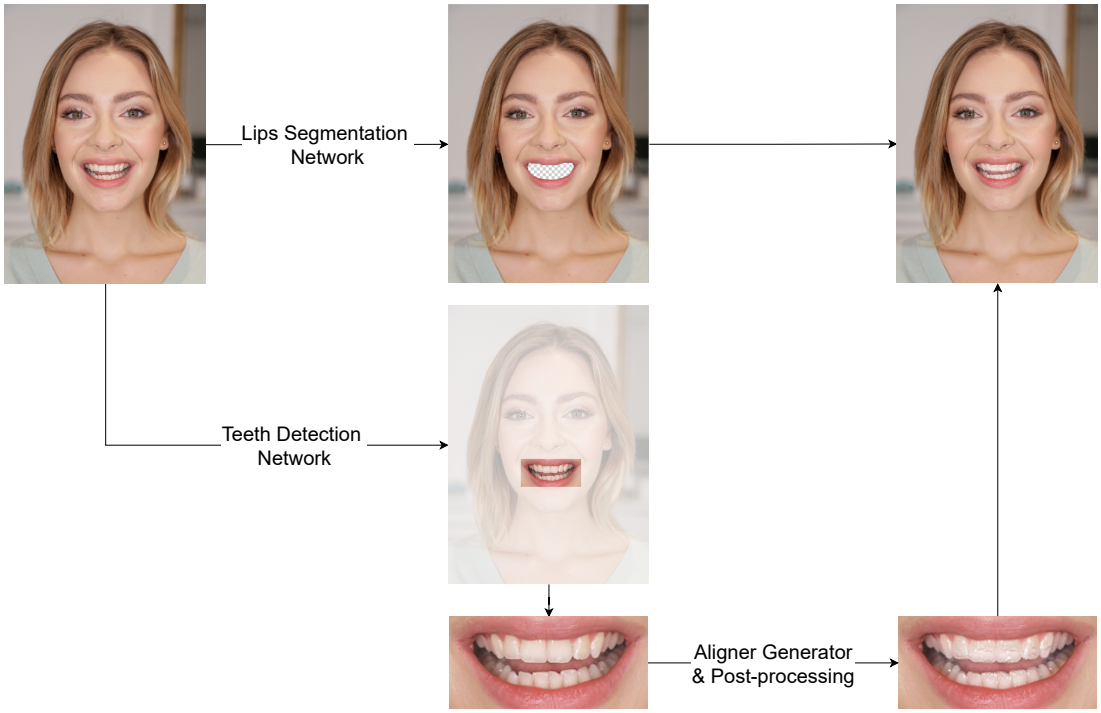



Figure 6.1: An overview of the aligner generator method with preprocessing steps.

Figure 6.1 demonstrates the process of generating aligners on a patient's portrait. The orthodontist firstly takes a portrait photo of the patient. To maximize the technique's benefit, the patient is asked to smile while trying to show as many teeth as possible. Naturally, when a person smiles, most of the upper teeth are visible, while not much of the lower teeth are seen. However, in this application, we would like to show the patient how aligners would look on teeth from upper and lower arches. Therefore, it is important to remind the patient to make both arches visible when shooting the portrait. Proper lighting condition is highly recommended.

Afterwards, the photo is submitted to our server to be treated. A segmentation network locates the oral region in the photo and erases it from the photo (top-middle of Figure 6.1).

Meanwhile, an object detection network detects teeth in the photo, draws a rectangle containing them and crop it out (bottom-middle of Figure 6.1). The cropped image is then sent to an image-to-image translation network which generates details of aligners on the teeth. We apply the color-transferring technique [133] for adopting the original color of the original image to the generated one. Next, the generated image is combined with the treated portrait to produce the final result (top-right of Figure 6.1), which is then sent back to the orthodontist to be shown to the patient. Having an estimated vision of how the appliance will look like can help the patient be more confident with his/her choice of treatment.

In this work, we train image-to-image translation models to generate transparent aligners on images of teeth. The model succeeds in generating realistic details of aligners on images of resolution up to $1024 \times 1024$. Our work is the first research ever working on generating aligners on teeth. We also propose a post-process step to keep transfer as many domain-irrelevant attributes as possible from the original images to the outcome.

### 6.3.2 CycleGAN

CycleGAN [188] is a deep learning model which is widely used for unsupervised image-to-image translation tasks such as horses↔, edges↔shoes, satellite-view↔map and so on.

#### Architecture

The method is known for the cycle-consistency loss, enabling the neural network to learn from unpaired samples. This is a suitable approach for our case where it is impossible to have pairs of oral images with the same pose and distance from the smartphone camera. We train CycleGAN to learn the mapping between two domains: no-aligners and with-aligner, as displayed in Figure 6.2.

For each training step, the network is fed with two input images: $x_A$ from domain $A$ (not wearing aligners) and $x_B$ from domain $B$ (wearing aligners). The image $x_A$ serves as an input of $G_{A\to B}$ which learns to map images from domain $A$ to domain $B$. The output of $G_{A\to B}$ is defined as:

$$x_{A\to B} = G_{A\to B}(x_A) \tag{6.1}$$

The generated image is then sent to $G_{B\to A}$ to be reverse the transformation and result an image $recon_A$ that belongs to the domain $A$:

$$\hat{x_A} = G_{B\to A}(x_{A\to B}) = G_{B\to A}(G_{A\to B}(x_A)). \tag{6.2}$$

Similarly, the input image $x_B$ from domain $B$ is transformed by the generator $G_{B\to A}$:

$$x_{B\to A} = G_{B\to A}(x_B) \tag{6.3}$$

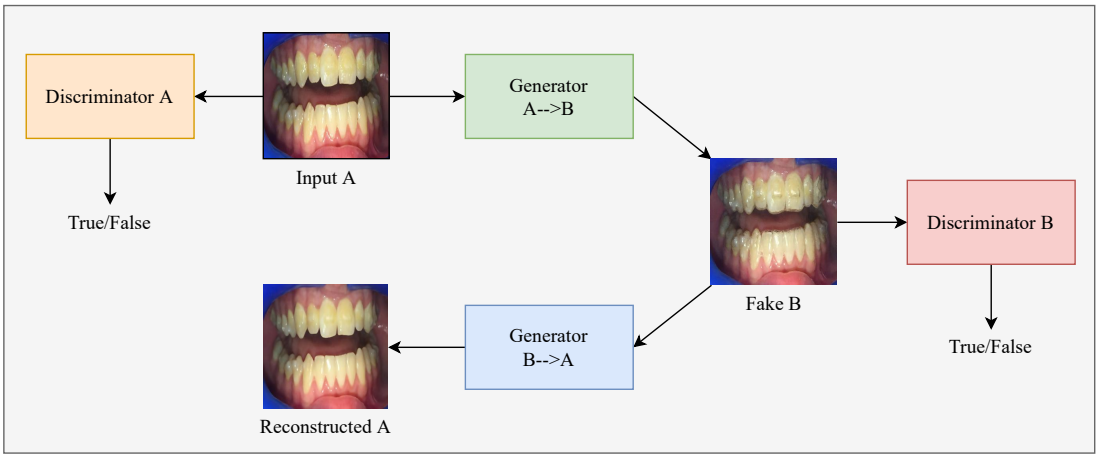
and reconstructed by the generator $G_{A\to B}$:

$$\hat{x_B} = G_{A\to B}(x_{B\to A}) = G_{A\to B}(G_{B\to A}(x_B)). \tag{6.4}$$

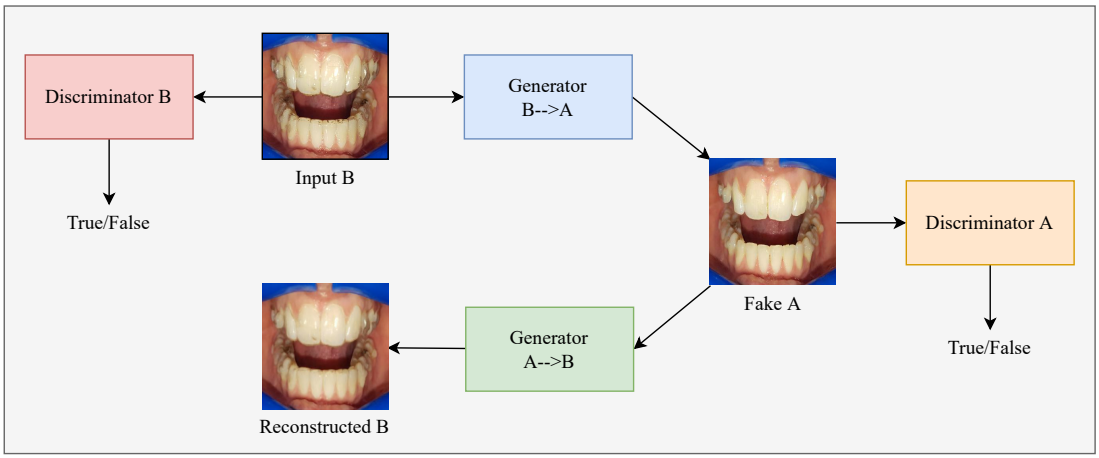
#### Training Losses

The model is trained with an adversarial loss which encourages the realism of the generated images and cycle-consistency loss to keep domain-irrelevant features after the transformation between two domains.

CycleGAN contains two discriminators for the two domains. For generating realistic images of domain $B$, we have discriminator $D_B$ as a classifier that learns to distinguish real images $x_B$ and generated images $x_{A\to B}$. The first adversarial objective is defined as:




**(a)**




**(b)**

Figure 6.2: The overview of CycleGAN architecture. The model is trained to generate and remove aligners on dental images.

$$L_{GAN}(G_{A\to B}, D_B, A, B) = \mathbb{E}_{x_B\sim p_{data}}(x_B)[logD_B(x_B)] + \mathbb{E}_{x_B\sim p_{data}}(x_B)[log(1 - D_B(G_{A\to B}(x_A)))] \tag{6.5}$$

Similarly, the adversarial objective of the discriminator $D_A$ which learns to detect generated samples $G_{B\to A}(x_B)$ from the training samples $x_A$ is expressed as:

$$L_{GAN}(G_{B\to A}, D_A, B, A) = \mathbb{E}_{x_A\sim p_{data}}(x_A)[logD_A(x_A)] + \mathbb{E}_{x_A\sim p_{data}}(x_A)[log(1 - D_A(G_{B\to A}(x_B)))] \tag{6.6}$$

Last but not least, the cycle consistency loss of the model is calculated as follow:

$$L_{cyc}(G_{A\to B}, G_{B\to A}) = \mathbb{E}_{x_A\sim p_{data}(x_A)}[||G_{B\to A}(G_{A\to B}(x_A)) - x_A||_1] + \mathbb{E}_{x_B\sim p_{data}(x_B)}[||G_{A\to B}(G_{B\to A}(x_B)) - x_B||_1]. \tag{6.7}$$

The ultimate training objective of the model is defined as:

$$L_{CycleGAN} = L_{GAN}(G_{A\to B}, D_B, A, B) + L_{GAN}(G_{B\to A}, D_A, B, A) + \lambda L_{cyc}(G_{A\to B}, G_{B\to A}). \tag{6.8}$$

For increasing the efficiency of the training, we apply the least-squares loss [113] at the place of the negative log-likelihood. Least-square loss is becoming more and more popular in the training of GANs for its stability. Particularly, $G_{A\to B}$ and $G_{B\to A}$ are trained to minimize the two objectives, respectively:

$$\mathbb{E}_{x_A\sim p_{data}(x_A)}[(D_B(G_{A\to B}(x_A)) - 1)^2] \tag{6.9}$$

and

$$\mathbb{E}_{x_A \sim p_{data}(x_B)}[(D_A(G_{B \to A}(x_B)) - 1)^2] \tag{6.10}$$

The least-square loss is applied for the pair of discriminators as well. Therefore, $D_B$ is trained to minimize

$$\mathbb{E}_{x_B \sim p_{data}(x_B)}[(D_B(x_B) - 1)^2] + \mathbb{E}_{x_A \sim p_{data}(x_A)}[(D_B(G_{A \to B}(x_A)))^2] \tag{6.11}$$

while $D_A$ learns to minimize

$$\mathbb{E}_{x_A \sim p_{data}(x_A)}[(D_A(x_A) - 1)^2] + \mathbb{E}_{x_B \sim p_{data}(x_B)}[(D_A(G_{B \to A}(x_B)))^2]. \tag{6.12}$$

### Training Details

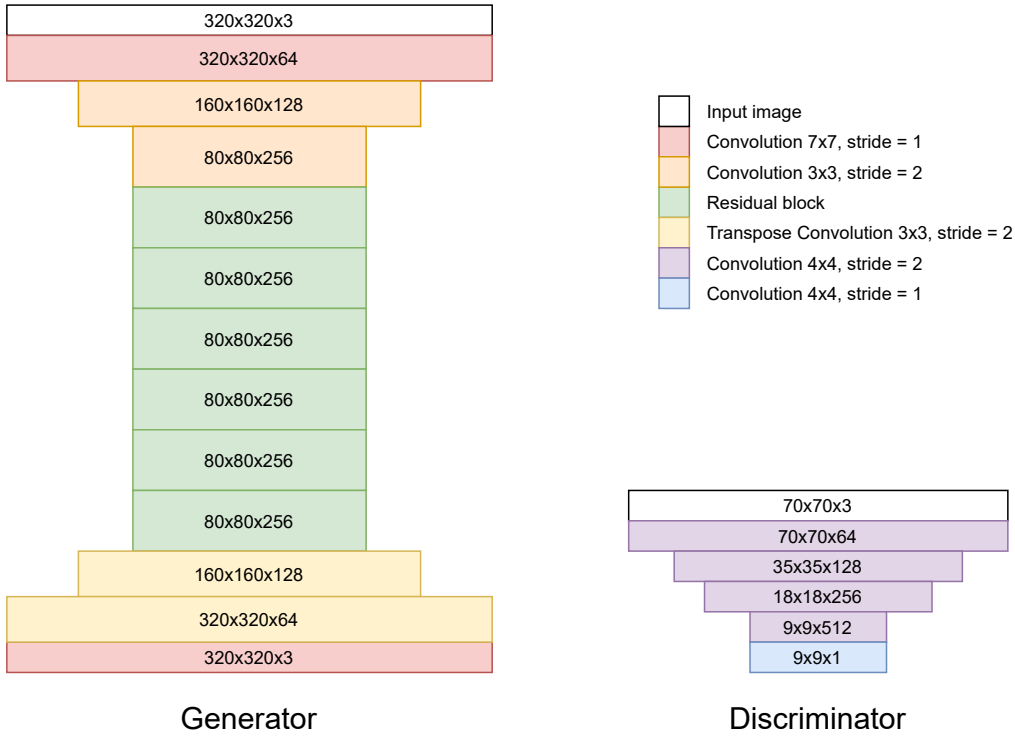



Figure 6.3: The architecture in details of the generators (left) and discriminators (right) in CycleGAN. For our experiment, the input image is in RGB with the size of 320x320.

We follow the architecture of CycleGAN in [188]. Figure 6.3 displays the architecture in detail of CycleGAN sub-networks. The generators consist of three convolutional layers, nine residual blocks, two fractionally-strided convolutional layers (stride 1/2), and one final convolutional layer that outputs an RGB image. Convolutions are followed by instance normalization. With this structure, the inputs and outputs of the generator always have the same size. For maximizing 11 GB of frame buffer on our device GTX 1080 Ti graphic card [124], the model is trained with images of size $320 \times 320$.

The discriminator does not take the whole image as input but randomly picks patches of size $70 \times 70$ for the prediction. Experiments of Zhu et al. in [188] show that the discriminators perform well on those patches as on full-size images. Plus, training discriminators on those patches consume less computation resource and training time than training with full-size images. The discriminator is trained using the last 50 generated images rather than only the last one. Adam is applied as the optimizer for the training with the initial learning rate of 0.0002, and decays to zero from the $101^{th}$ epoch to the $200^{th}$ epoch where the training ends. We apply the batch size of 1 and $\lambda$ (Equation 6.8) of 10 to emphasize the cycle consistency.

### 6.3.3 StarGANv2

StarGANv2 [27] is another unsupervised image-to-image translation model.

#### Architecture

Instead of training one generator and one discriminator for each mapping, only one generator $G$ and one discriminator $D$ are used for all the translations. It has two more networks than CycleGAN: a mapping network $F$ and a style encoder $E$. However, $F$, $E$ and $D$ share early layers, so StarGANv2, in fact, contains fewer training parameters than CycleGAN. In other words, StarGANv2 consumes less memory and is less likely to be over-fitted.

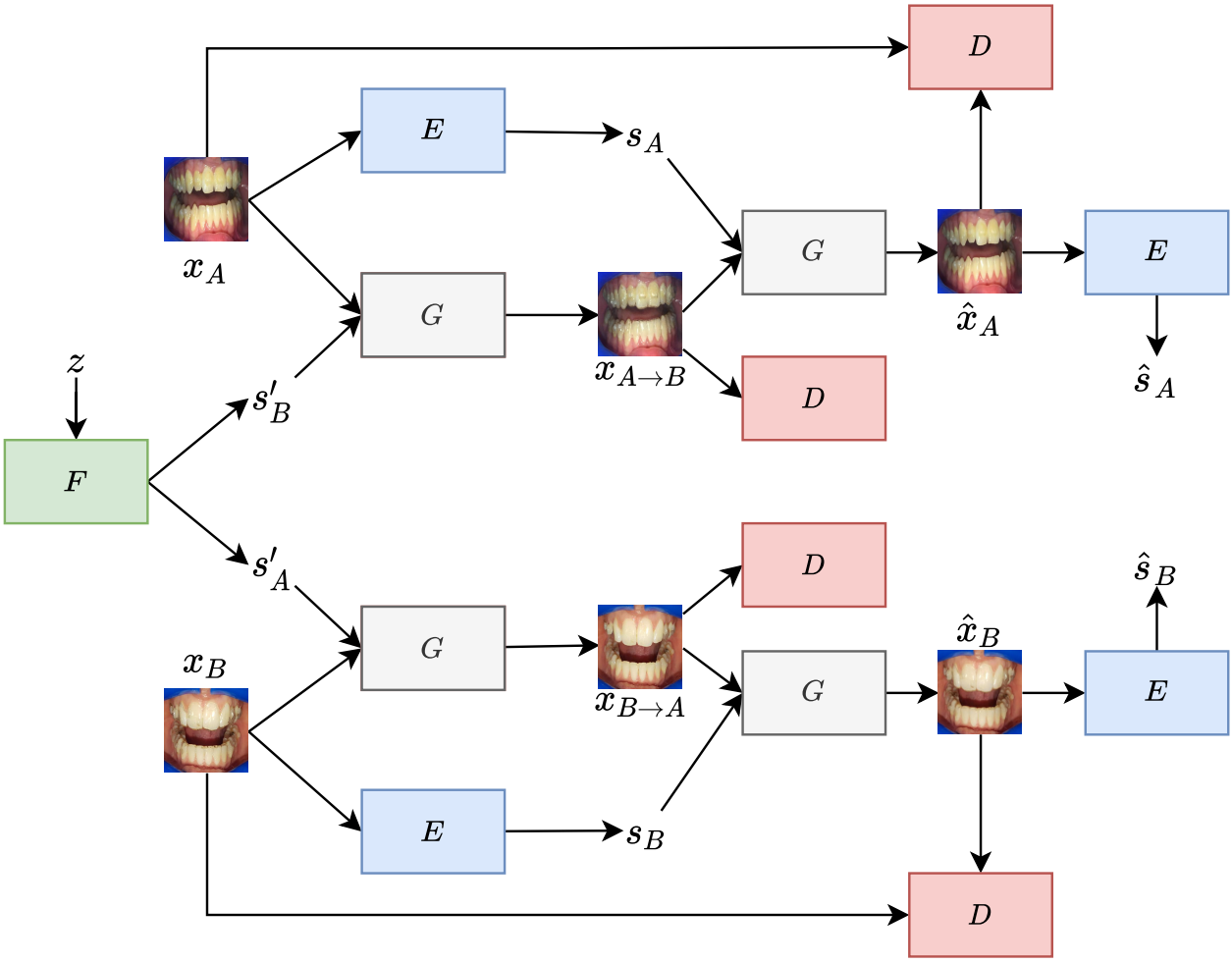



Figure 6.4: The overview of StarGANv2 architecture. The model is trained to generate($A \to B$) /remove ($B \to A$) aligners on dental images.

Figure 6.4 illustrates the architecture of StarGANv2. Given an input image $x_A$ and a domain-specific style code $s_B$, the generator $G$ generates an output $x_{A\to B}$. The input image of $G$ can come from the real dataset or being generated by $G$ itself, while the style code can be sent from the mapping network $F$ or the style encoder $E$. The style code $s_B$ can be generated by the mapping network $F$ or the style encoder $E$. It is injected into $G$ using adaptive instance normalization (AdaIN), widely used in style transfer models for simplicity and efficiency. Using domain-specific style code is one of the biggest differences of StarGANv2 to the original of StarGAN, [28] where a domain label is used. According to [27], a single label does not contain enough reference information, so it limits the diversity of generated images. Therefore, using a domain-specific style code assures that the generated image resembles the target domain and increases the output diversity at the same time. Containing multiple branches, the mapping network $F$ generates a style code for each domain given a random latent code $z$. This architecture enables $F$ to learn the representations of both domains. The style encoder $E$ learns

to extract style from any given image. It also provides a style code for each domain thanks to the multiple-branches architecture like in $F$. Using $E$ and $G$, we can learn to transfer the style of one image to another. In the scope of this research, we do not implement this task. In this work, we have two domains: no-aligners and with-aligners, so $E$ and $F$ have two branches representing the two domains. Like any other GANs-based model, StarGANv2 includes a discriminator $D$, which distinguishes between real and fake images. Like $E$ and $F$, $D$ also consists of multiple branches to deal with multiple domains' images.

In total, StarGANv2 consists of four networks, just like CycleGAN, to learn the mappings between two domains of data. However, one main advantage of StarGANv2 to CycleGAN is that $E$, $F$ and $D$ share most of their convolutional layers, leading to a smaller amount of learning parameters and lower computation cost. This characteristic enables us to train StarGANv2 on high-resolution images.

### Training Details

For our experiments, all training losses of StarGANv2 are kept just like explained in Section 3.6.3: Equations 3.36, Equations 3.37, Equations 3.38, Equations 3.39 and Equations 3.40. Weights in Equations 3.40 are set as $\lambda_{style} = \lambda_{diverse} = \lambda_{cycle} = 1$ for all trainings.

The StarGANv2 network is trained on the same dataset as CycleGAN. Thanks to the compacted architecture, we can train the model on a much larger resolution compared to CycleGAN. It is trained at three different resolutions: $256 \times 256$, $512 \times 512$ and $1024 \times 1024$. We try to maximize the batch size to make use of the memory of our computing device, GTX 1080 Ti. The batch size settings for the three resolutions are respectively 4, 2 and 1. The model is trained during 200 thousand iterations to make sure that it is well converged. As the diversity of outputs is not our critical goal, we decay $\lambda_{diverse}$ to 0 over the first 100 thousand training iterations. Then, $\lambda_{diverse} = 0$ until the end of the training so that the model can focus more on improving the quality of output images. Other training hyperparameters are set like in the original work of StarGANv2. The classical Adam optimizer [87] is applied with $\beta_1 = 0$ and $\beta_2 = 0.99$. $G, D$ and $E$ are trained with a learning rate of $10^{-4}$ while the learning rate of $F$ is $10^{-6}$. Parameters of the model are initialized with Kaiming initialization (or He initialization) [61]. Biases related to the AdaIN are set to 1 while other biases are set to 0. We apply $R_1$ regularization with $\gamma = 1$ and the non-saturating GAN loss [56] to stabilize the training.

Figure 6.5 displays in detail the configuration of all StarGANv2 sub-networks.

StarGANv2 generator (Figure 6.5 top-left) starts and ends with a convolutional layer, with 12 residual blocks [62] at the middle. The first four blocks are followed by average-pooling operations to downsample the width and height of the tensor. Then, the tensor goes through four blocks without downsampling. The four last residual blocks are followed by upsampling operations to recover the size of the input. Instance normalization (Section 3.3.2) is applied for the first six residual blocks while applying adaptive instance normalization (aka. AdaIN, Section 3.3.4) for the other blocks. Through these AdaIN layers, the generator injects a style code into the transformation so that the given style code influences the output image style.

The mapping network (Figure 6.5 top-right) is a (multilayer perceptron) MLP that begins with four fully connected layers and then splits into two domain-specified branches. Each branch consists of four fully connected layers. The input latent is a vector with 16 digits sampled from the Gaussian distribution. All layers in the mapping networks have the dimension of 512 except the final layer of each branch that produces the style code, which is a vector of 64 dimensions. Pixel normalization and feature normalization are not applied in our model because they do not improve the performance of StarGANv2 according to [27].

The discriminator (Figure 6.5 bottom-left) in StarGANv2 has the multi-task architecture. Like the mapping network, it also has one branch for each domain. It contains convolutional layers, six residual blocks with pooling layers for downsampling, another convolutional layer before divided into two branches. Each contains a fully connected layer. Each fully connected layer outputs a single neuron that holds the prediction true/false for one domain. The last resid-

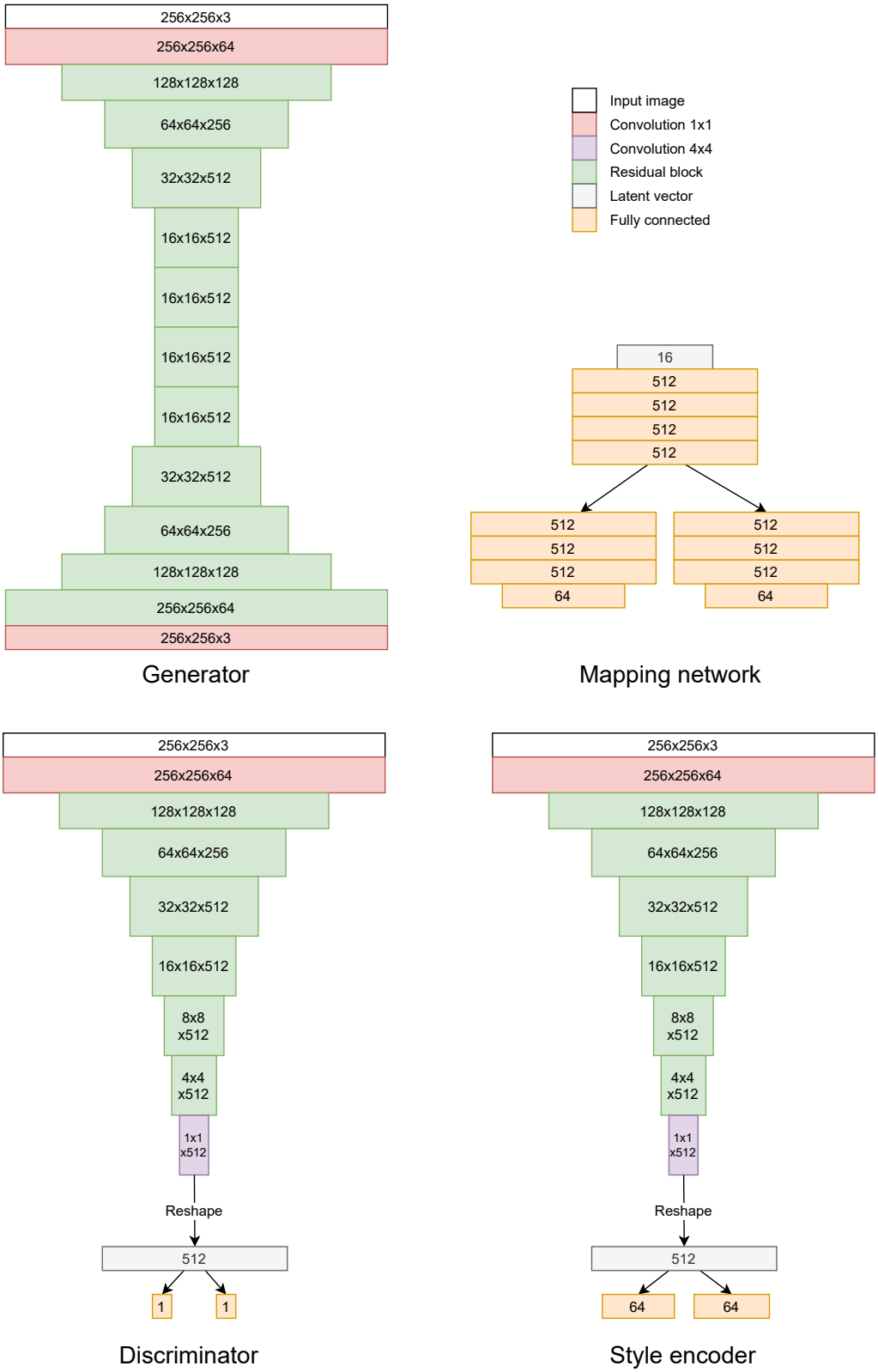



Figure 6.5: The architecture of each sub-network in StarGANv2: Generator (top-left), mapping-network (top-right), discriminator (bottom-left) and style encoder (bottom-right). In this example, the input image is in RGB with the size of $256 \times 256$.

ual block and the $4 \times 4$ convolutional are followed by leaky ReLU [107]. Feature normalization and PatchGAN technique in CycleGAN [188] are reported to not increase the discriminator's performance according to [27] so we do not apply them.

The style encoder has an identical structure as the discriminator with convolutional layers and residual blocks, except for the two branches' output layers. For each domain, it produces a domain-specified style code of length 64.

## 6.4 Evaluation

We measure the performance of our models on two factors: realism and diversity of the generated images. The two metrics are Fréchet Inception Distance (FID, Section 3.5.4) and the learned perceptual image patch similarity (LPIPS, Section 3.5.3) are chosen for the evaluation. The evaluation is done using 800 images in the validation set.

### 6.4.1 Fréchet Inception Distance (FID)

FID computes the distance between the set of generated images and the set of real images in the feature space. A small FID value means that the generated images and real images look alike, which means generated images are realistic. Each image in the two sets is represented by a feature vector computed by feeding the image to the Inception v3 network pre-trained on ImageNet and getting the output at its last pooling layer.

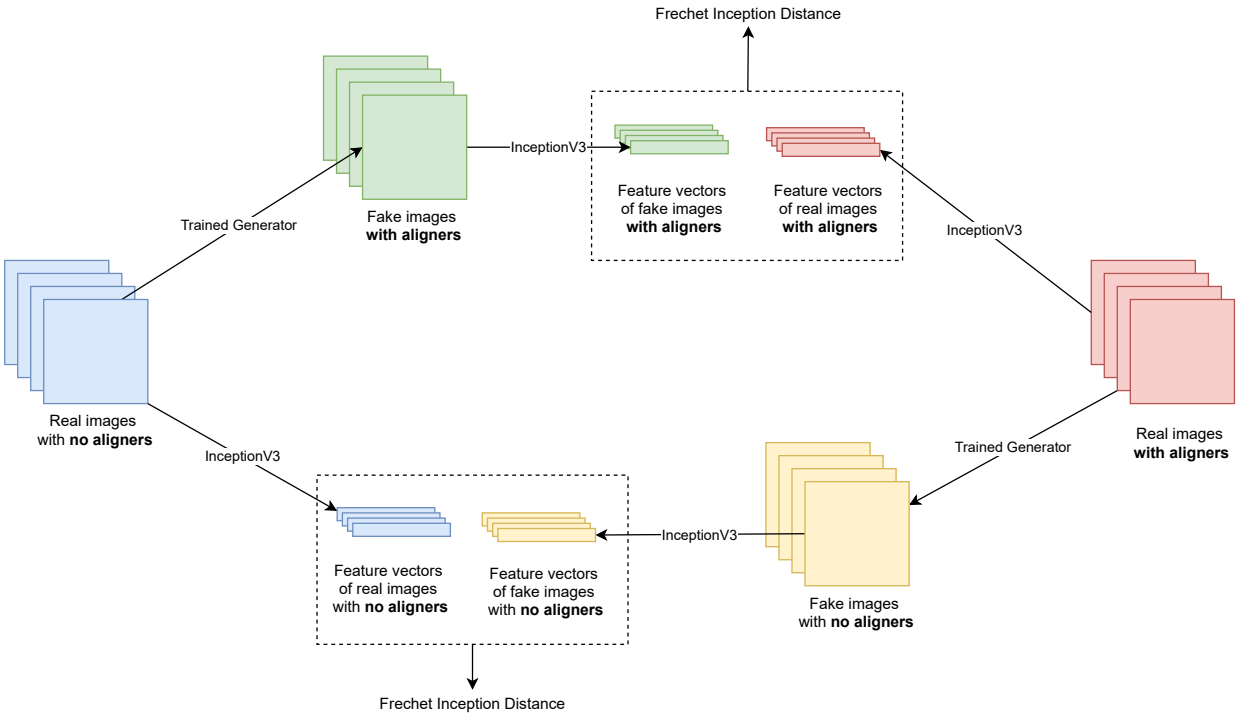



Figure 6.6:

Figure 6.6 explains how we generate images and computing FID from them. For each image in the validation set, we generate a fake image for the opposite domain, compute the feature vectors of those generated images, and compare them with the feature vectors of the target domain's real images. In this test, note that from one image in the validation set, we generate only one output. Therefore, the numbers of fake images generated by each method are equal, making the FID test fairer.

### 6.4.2 Learned Perceptual Image Patch Similarity (LPIPS)

LPIPS computes the distance in terms of visual features between two given images. The features are computed using a pre-trained CNN. In our experiments, we adopt an AlexNet model that is trained for the classification task on ImageNet. To measure the diversity of our models' outputs, we compute the LPIPS between all the possible pairs among all generated images. As our validation set contains 400 images, the amount of pairs being used to compute LPIPS is equal to

$$C(400, 2) = \frac{400!}{2!(400-2)!} = 79800. \tag{6.13}$$

Then, the mean value of all the LPIPS of all pairs is considered the final LPIPS. A higher value of LPIPS refers to a better diversity.

## 6.5 Results & Discussions

In this section, we compare the performance of the two methods CycleGAN and StarGANv2, on two transformation tasks: no-aligner→with-aligner and with-aligner→no-aligner. Using FID and LPIPS metrics, we measure the performance of the two networks in terms of image quality and diversity.

### 6.5.1 Image Quality

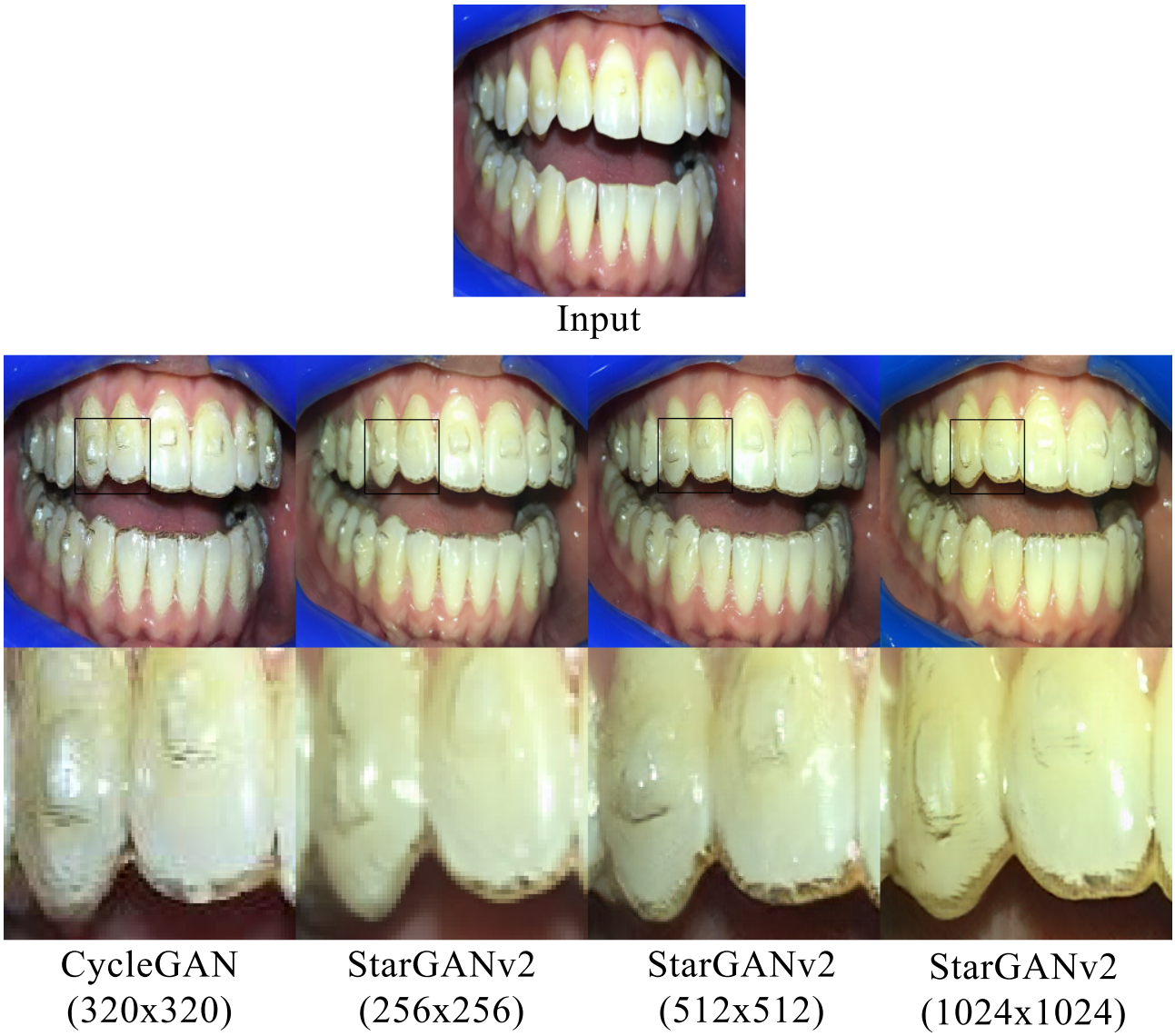



Figure 6.7: Qualitative comparison between outputs of CycleGAN and StarGANv2 at multiple resolutions for the task no-aligner→with-aligner.

Figure 6.7 shows the qualitative comparison of CycleGAN and StarGANv2. As mentioned before, $320 \times 320$ is the maximum size that our device can train CycleGAN, so we choose to train the network on this resolution to benefit the capacity. StarGANv2, for technical reasons, accepts only resolutions that are powers of 2 and consume less memory than CycleGAN, so we train the network on three different resolutions: $256 \times 256$, $512 \times 512$ and $1024 \times 1024$. As can be seen from Figure 6.7, output images generated by both methods look quite realistic. The

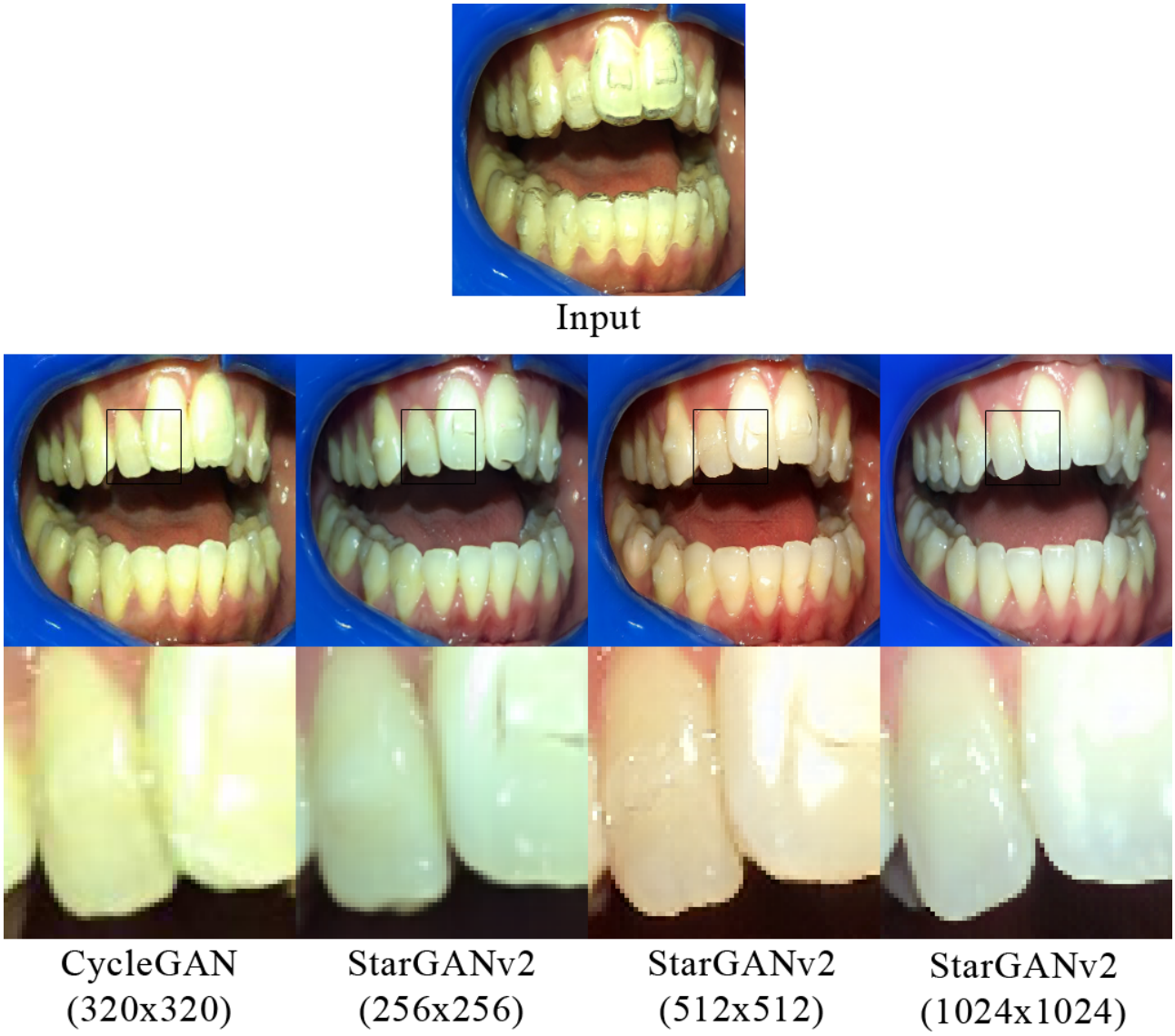



Figure 6.8: Qualitative comparison between outputs of CycleGAN and StarGANv2 at multiple resolutions for the task with-aligner→no-aligner.

figure also shows that StarGANv2 successfully generates realistic images even at high resolution ($1024 \times 1024$) with well-generated details. One can also notice that the output of CycleGAN at the resolution of $320 \times 320$ contains tiny artifacts making it less realistic than the outputs of StarGANv2 even at the resolution of $256 \times 256$. We observe a similar trend in Figure 6.8 when the two networks are trained for removing aligners from images where the patient is wearing aligners. Taking a deeper look at the results, we can see that different models give slightly different shapes for the teeth number 12. This is because, in the input, the edge of these teeth is not well displayed. Intuitively, removing aligners is more difficult than adding them on teeth because we do not have a clear vision of the original shape of teeth because of aligners' presence. As the scope of this project, we focus on generating aligners on teeth, but improving the performance of StarGANv2 for the opposite task would be addressed in future work.

The two methods are also compared quantitatively using FID as the evaluation technique, displayed in Figure 6.9. As explained before, FID measures the difference between the generated images and the real ones. A good generative model should generate images that produce a small FID. According to the figure, StarGANv2 achieves a significantly lower FID at all resolutions compared to CycleGAN. We also observe that the FID of StarGANv2 raises with the image resolution. However, the FID of StarGANv2 at the size of $1024 \times 1024$ is still much lower than the score of CycleGAN at a much smaller size ($320 \times 320$).

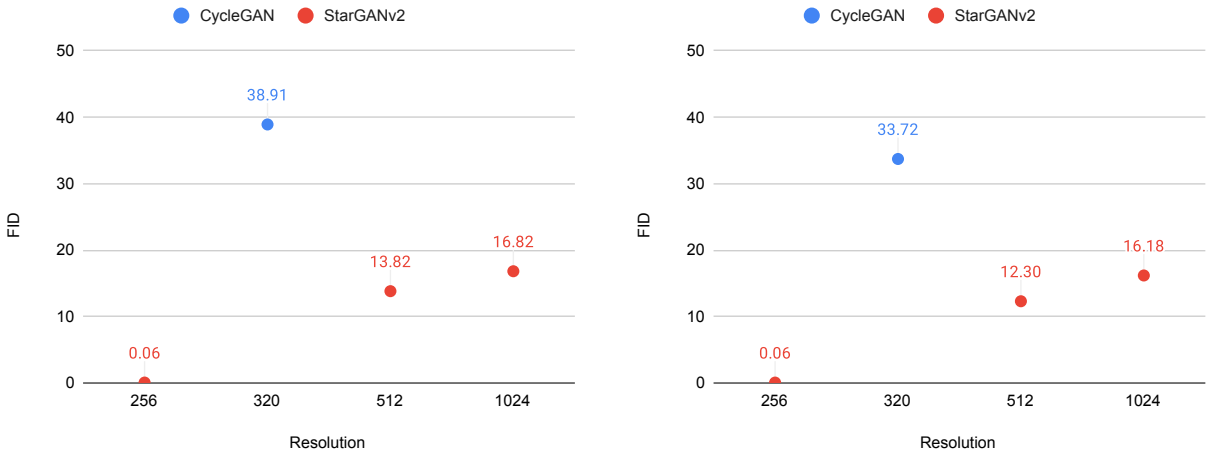



Figure 6.9: Comparison of the Fréchet Inception Distance (FID) between generated samples and real samples in the test set for the two methods CycleGAN and StarGANv2 (lower is better). The comparison is done on two tasks: *generating aligner* (left) and *removing aligner* (right).

### 6.5.2 Image Diversity

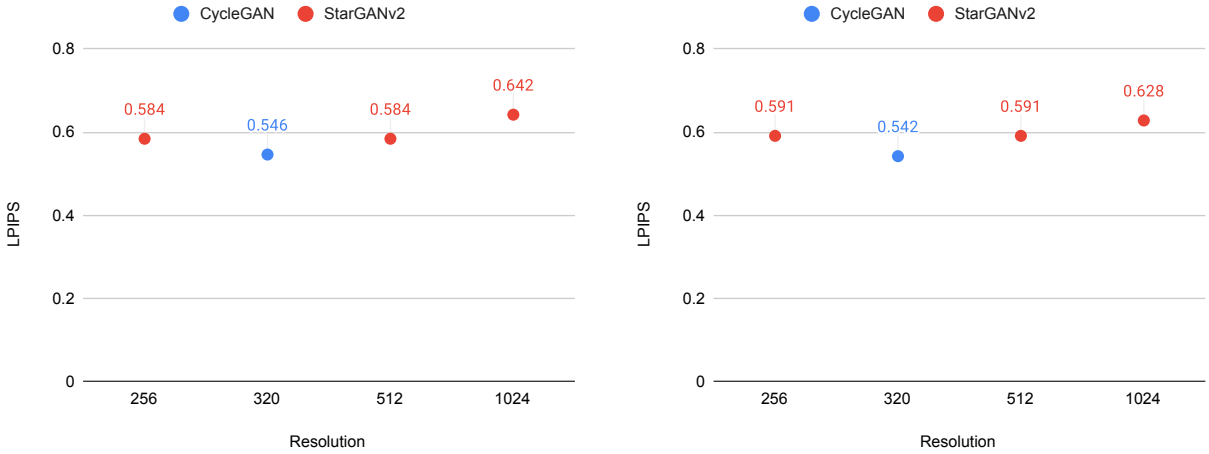



Figure 6.10: Comparison of the LPIPS between generated samples and real samples in the test set for the two methods CycleGAN and StarGANv2 (higher is better). The comparison is done on two tasks: *generating aligner* (left) and *removing aligner* (right).

We apply LPIPS as the metric to compare the ability to generate diverse outputs between CycleGAN and StarGANv2. Figure 6.10 display the comparison result. It is shown that StarGANv2 gets a lightly higher LPIPS score at all resolutions than CycleGAN at the image size of $320 \times 320$. This improvement can be explained by the diversity sensitive loss (Equation 3.38) used for training StarGANv2 but not in CycleGAN.

## 6.6 Conclusion

In this chapter, we apply image-to-image translation methods CycleGAN and StarGANv2 to generate aligner details on portraits that help orthodontic patients approximate how they will look wearing aligners. Our experiments show that both techniques perform well on dental images, just like other datasets such as horses↔zebras, edges↔shoes or edges↔handbags. We also point out that StarGANv2 uses computation resources more effectively so it can work

with images up to the resolution of 1024×1024 on our device GTX 1080Ti, while CycleGAN is limited at the size of $320 \times 320$. Not only uses memory effectively, but StarGANv2 also generates highly realistic images at the resolution of $1024 \times 1024$. For future work, it would be beneficial if the speed of inference can be reduced so that the output can be generated faster, even in real-time.

# Chapter 7

# Informative Multimodal Unsupervised Image-to-Image Translation

## 7.1 Introduction

Image-to-image translation can be described as the general problem of mapping an image from one domain to another. This seemingly simple approach is the foundation of many applications in computer vision, such as colorization [76], style transfer [50], super-resolution [37], denoising, inpainting [178]. Moreover, image-to-image translation has also been applied for data augmentation and achieved competitive results [119] [102] [70]. Based on the availability of data, the problem can be considered as supervised learning where the dataset contains paired samples, or unsupervised learning, where the dataset consists of two independent sets of images. This work focuses on the unsupervised image-to-image problem, which is more applicable due to its ease of obtaining data and more challenged in terms of training.

Unsupervised image-to-image translation leads us to the idea that an image in a domain can be translated into multiple images in the second domain, which means the translation can be multimodal. For example, in image colorization, one image can be colored in multiple ways. Some methods [96] [73] have been proposed to use a noisy vector as an additional input of the decoder. The style of the generated images can then be manipulated by changing the values of the style-vector. However, the style-vectors in existing methods are entangled, and the translated images are not interpretable as a result. Lacking control over features of the output can be problematic when important information is linked to these features. In the work of Cohen et al. [29], it is shown that CycleGAN was adding/removing tumors from images when transforming MRI images from Flair to T1, especially when there is an imbalance among classes in the training data. Therefore, learning to control the features of the translated images is essential. In this chapter, we propose some improvements on MUNIT [73] - a standard in the field of multimodal image translation - by applying the mutual information maximization technique. Our method, called InfoMUNIT, generates more diverse images and especially can manipulate their textural and structural features without requiring any annotation.

## 7.2 Methodology

Our objective is to translate images from a source domain A to a target domain B, and at the same time, to learn the representation of the target domain. Following the idea called partially shared latent space in [73], we assume that each image can be encoded as a content code containing general structural information and a style code which defines how the image looks like. In state-of-the-art methods, this style latent code is entangled. In this work, we disentangle this style code by maximizing the mutual information between this code and the generated image's features.

### 7.2.1 Network architecture

Let $x_A$ and $x_B$ be two images from domain $A$ and $B$ respectively. Our objective is to learn a function $F_{A\to B}$ that projects images from domain $A$ to domain $B$, $\hat{x}_{A\to B} = F_{A\to B}(x_A)$. This function can be decomposed into parts: the encoder and the generator. The encoder $E_A^c$ extracts the content code $c_A$ from the image. The content code is a matrix representing the content of an image independently of its style. The generator $G_B$ generates images in domain $B$ from an content code and a style code $s$ : $\hat{x}_{A\to B} = G_B(c_A, s) = G_B(E_A^c(x_A), [s', i])$. Since we want a one-to-many projection, a style code $s_B$ is inputted in the generator to introduce variability in the generated images. The style code $s$ is a vector created by concatenating two parts $s'$ and $i$ where $s'$ stores entangled style of the generated images, $i$ contains disentangled features of the generated images. $s'$ and $i$ are drawn from a normal distribution $N(0, I)$. The generator learns a function that links the points from a Gaussian distribution to the different ways to apply the style of domain B to a content code. In the same way, we define the function that projects images from domain $B$ to domain $A$ with generator $G_A$ and encoder $E_A^c$. Notice that the content space and style space are common to both domains. These generators project a couple of points from these common spaces to the image sub-spaces corresponding to their domain.

For the learning of these functions, we need to complete our architecture with autoencoders and discriminators. Autoencoders are used to reconstruct the original images from their decomposition into a content code and a style code. Let $E_A^s$ (resp. $E_B^s$) denote the encoder that extracts from an image of domain $A$ (resp. $B$) its style code $s_A = [s'_A, i_A]$ (resp. $s_B = [s'_B, i_B]$). The autoencoder of domain $A$ is therefore defined by $\hat{x}_A = G_A(E_A^c(x_A), E_A^s(x_A))$. Autoencoders are also used to reconstruct the content $\hat{c}_A = E_A^c(G_B(c_A, s))$ and style codes $\hat{s} = E_B^s(G_B(c_A, s))$. The discriminator $D_B$ is used to align the distribution of images produced by the generator $G_B$ with the distribution of original images from domain $A$. It is also used to disentangle the style variables contained in the vector $i$. In the same way, we define the autoencoders $\hat{x}_B = G_B(E_B^c(x_B), E_B^s(x_B))$, $\hat{c}_B = E_B^c(G_A(c_B, s))$, $\hat{s} = E_A^s(G_A(c_B, s))$ and discriminator $D_A$.

Figure 7.2 shows the complete architecture of InfoMUNIT. Each image is encoded by two encoders into a style code and a content code and reconstructed by a decoder (also called generator). To translate an image from a domain to another domain, we first extract its content code, combine it with a random style code, and send them both to the generator of the target domain. A part of the style code is used to store disentangled features of output images. We also train a pair of discriminators to distinguish between generated images and real images for each domain. The generators are also trained to maximize the mutual information between features being extracted by those discriminators and the disentangled part in the style code.

Figure 7.1 visualizes in detail the configuration of all sub-networks in MUNIT and InfoMUNIT. In terms of network architecture, our InfoMUNIT is not much different from the original MUNIT, except the discriminator. The content encoder and decoder form an autoencoder similar to the generator in CycleGAN [188] with convolutional layers for downsampling and upsampling with $3 \times 3$ residual blocks the middle. In MUNIT/InfoMUNIT, upsampling is done by $2 \times 2$ nearest-neighbor upsampling followed by a $5 \times 5$ convolution. Style encoder is a new point in the architecture of MUNIT compared to CycleGAN [188] and UNIT [73]. It compresses the style of an input image to an 8-dimensional vector using a series of $7 \times 7$ and $4 \times 4$ convolutional layers. The sub-network also applies a global average pooling layer [100] before the final fully-connected layer. An advantage of global average pooling over fully connected is that it takes fewer learning parameters and at the same time increases the robustness of the architecture to input sizes. Each feature map is compressed into a single neuron regardless of its width and height. The style code is injected into the decoder feature maps using AdaIN (Equation 3.8), which works on the feature maps' mean and variance. The technique is suitable to make changes in style without changing too much structural information of the output image. The main difference in architecture between InfoMUNIT and MUNIT is the discriminator. While the MUNIT discriminator simply consists of residual blocks and a $1 \times 1$ convolutional

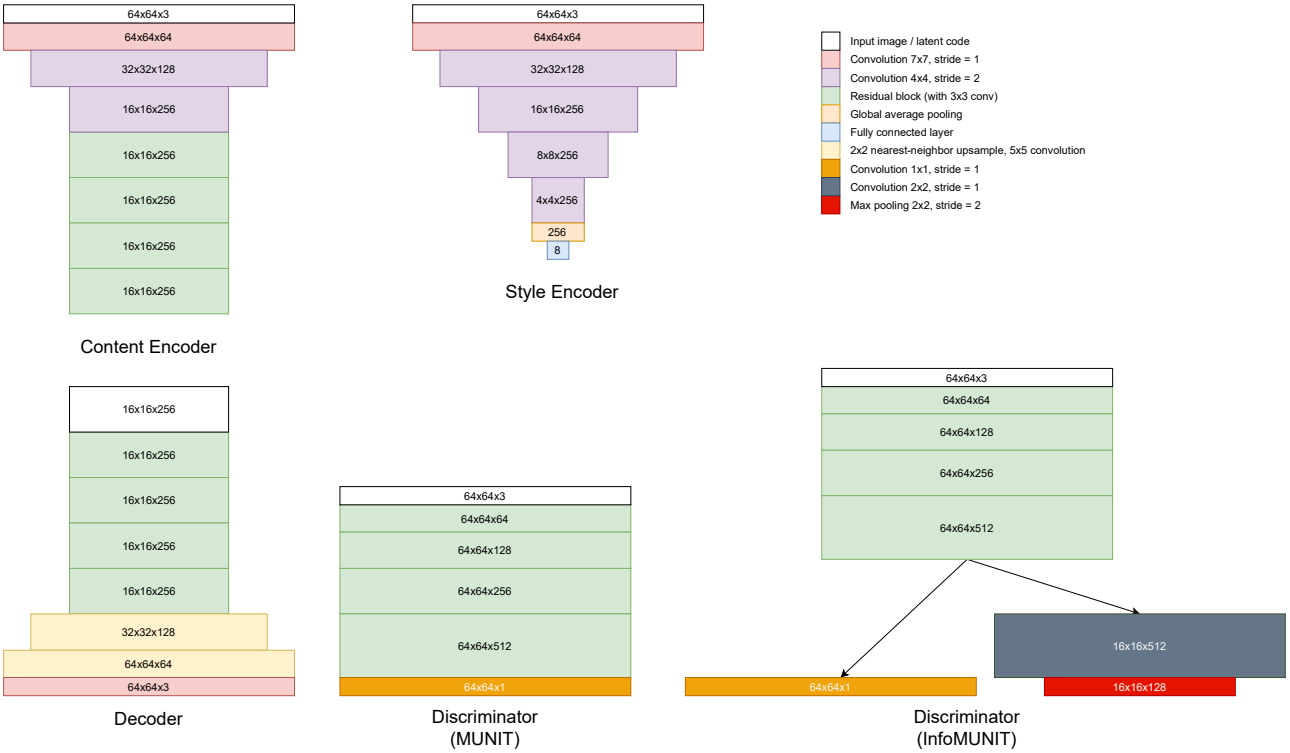



Figure 7.1: The architecture of each sub-network in MUNIT and InfoMUNIT including Content Encoder (top-left), Style Encoder (top-middle), Decoder (bottom-left), Discriminator in MUNIT (bottom-middle) and Discriminator in InfoMUNIT (bottom-right).

layer at the end, the InfoMUNIT discriminator has an additional branch that represents the $Q$ distribution (Equation 7.9). Our goal is to train the decoder to learn to preserve a piece of information stored in the style code (output of the style encoder) in $Q$ (the second output of the discriminator).

### 7.2.2 Model learning

Our model's training consists of minimizing a combination of reconstruction losses and adversarial losses and maximizing the variational mutual information.

Like most autoencoder-based architecture, the encoders $E_A^C$ and $E_A^S$ compress input images to content code and style code while the generator $G_A$ takes them to reconstruct the original image from domain $A$. The image reconstruction loss $\mathcal{L}_{rec}^{x_A}$ makes sure the encoder and decoder inverse each other. $L_1$ loss is chosen for the image reconstruction as it usually obtains well the sharpness of the reconstructed image. For the same reason, we have similar reconstruction losses for content code $\mathcal{L}_{rec}^{c_A}$ and style code $\mathcal{L}_{rec}^{s_A}$.

$$\mathcal{L}_{rec}^{x_A} = \mathbb{E}_{x_A \sim p(x_A)}[\| G_A(E_A^c(x_A), E_A^s(x_A)) - x_A \|_1] \tag{7.1}$$

$$\mathcal{L}_{rec}^{x_B} = \mathbb{E}_{x_B \sim p(x_B)}[\| G_B(E_B^c(x_B), E_B^s(x_B)) - x_B \|_1] \tag{7.2}$$

$$\mathcal{L}_{rec}^{c_A} = \mathbb{E}_{c_A \sim p(c_A), s\sim\ p(s)}[\| E_B^c(G_B(c_A), s) - c_A \|_1] \tag{7.3}$$

$$\mathcal{L}_{rec}^{c_B} = \mathbb{E}_{c_A \sim p(c_B), s \sim p(s)}[\| E_A^c(G_A(c_B), s) - c_B \|_1] \tag{7.4}$$

$$\begin{aligned}\mathcal{L}_{rec}^{s} = &\mathbb{E}_{c_A \sim p(c_A), s \sim p(s)}[\| E_B^s(G_B(c_A), s) - s \|_1] \\ &+ \mathbb{E}_{c_B \sim p(c_B), s \sim p(s)}[\| E_A^s(G_A(c_B), s) - s \|_1]\end{aligned} \tag{7.5}$$

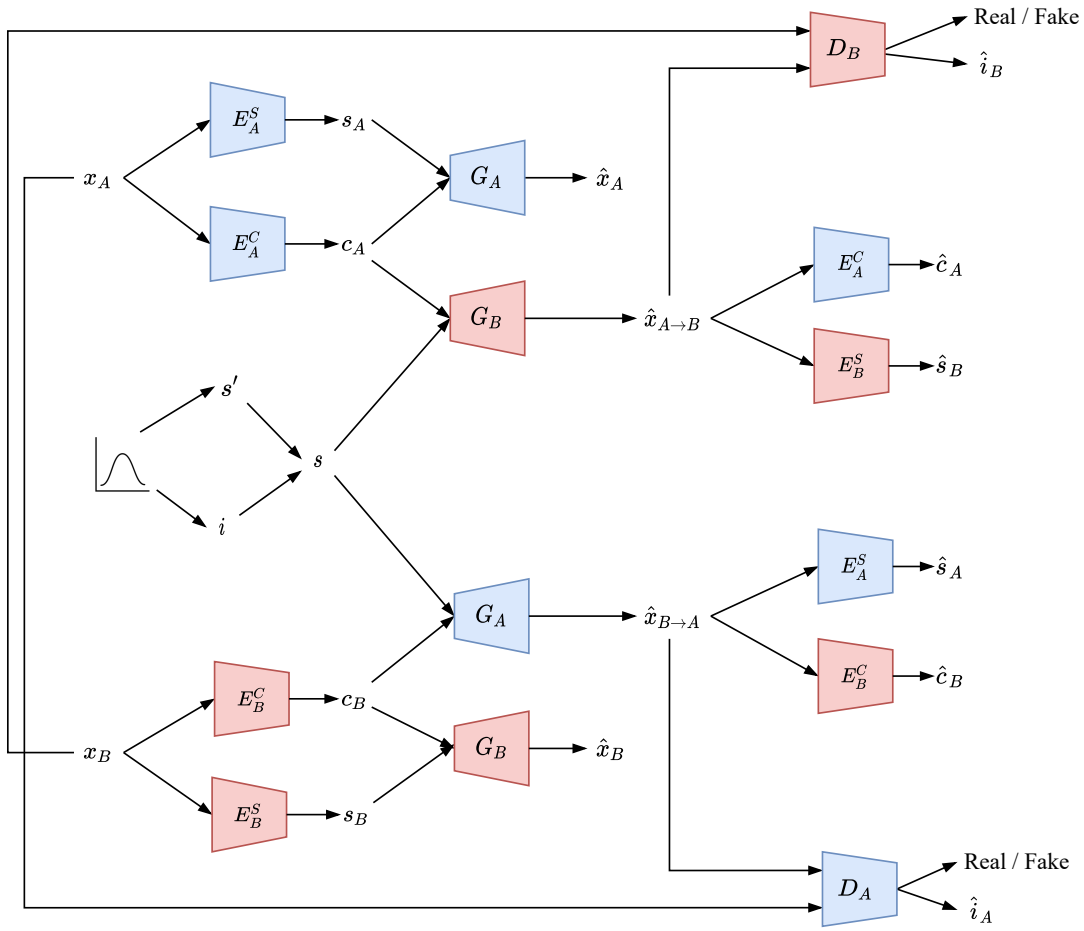



Figure 7.2: Overview of InfoMUNIT.

where $p(x_A)$ (resp. $p(x_B)$) is the distribution of images from domain $A$ (resp. $B$), $p(c_A)$ (resp. $p(c_B)$) is the distribution of content code extracted from images from domain $A$ (resp. $B$), and $p(s)$ is the distribution of style code that is the unit Gaussian distribution $N(0, I)$. Note that the distributions $p(c_A)$ and $p(c_B)$ are unknown and the learning set do not contains examples of $c_A$ and $c_B$., we need there fore to generate $c_A$ and $c_B$ samples from the encoders and training images $c_A = (E_A^c(x_A))$ and $c_B = (E_B^c(x_B))$.

The objective of the adversarial losses associated with the discriminators is to align the distributions of the real images with the distribution of the generated images. Like in the GAN, the discriminators try to predict if an image is a real one or an artificial image produced by the generator. When the generators are frozen, the generators try to fool the discriminators into generating images close to the real ones. The adversarial losses are defined by ;

$$\begin{aligned}\mathcal{L}_{adv}^A = \mathbb{E}_{x_B\sim p(x_B), s\sim p(s)}[log(1 - D_A(G_A(E_B^C(x_B); s)))] \\ + \mathbb{E}_{x_A\sim p(x_A)}[logD_A(x_A)]\end{aligned} \tag{7.6}$$

$$\begin{aligned}\mathcal{L}_{adv}^B = \mathbb{E}_{x_A\sim p(x_A), s\sim p(s)}[log(1 - D_B(G_B(E_A^C(x_A); s)))] \\ + \mathbb{E}_{x_B\sim p(x_B)}[logD_B(x_B)]\end{aligned} \tag{7.7}$$

where the output of the discriminator $D_A(x)$ (resp. $D_B(x)$ ) is the probability that the image $x$ is a real image from the domain $A$ (resp. $B$).

Inspired by the idea of InfoGAN [26], we want a part of the style code to be disentangled features of the output to control and improve the diversity of the translated images. The style code is split into two parts $s = [s', i]$. To encourage the subvector $i$ to represent the output's disentangled features, we maximize the mutual information between $i$ and the generated images.

$$\mathcal{I}(i, G_B(c_A, [s, i])) \quad \text{and} \quad \mathcal{I}(i, G_A(c_B, [s, i])) \tag{7.8}$$

In practice, maximizing this mutual information is not achievable without access to the distribution $P(i|x)$, which is not available in our case. However, according to [9], we can define an additional distribution $Q(i|x)$ as an approximation of $P(i|x)$, and get a lower bound of the mutual information term. Thus we have:

$$\mathcal{I}(i, G_B(c_A, [s, i])) \geq L_{mi}(G_B, Q_B) = \\ \mathbb{E}_{i\sim p(i), x_{A\to B}\sim P(G_B(c_A,[s',i]))}[logQ_B(i|x_{A\to B})] \tag{7.9}$$

Where $p(i)$ is a normal distribution and $P(G_B(c_A, [s', i]))$ is the distribution of the images generated by $G_B$ with the style vector $[s', i]$. In practice, $Q_B$ shares the same layers of the discriminator $D_B$ as they both extract features from $G_B(c_A, [s', i])$ . $Q_B$ is implemented as a secondary output of the discriminator $D_B$ that is notes $\hat{i}$. This means the closer the vector $i$ and predicted vector $\hat{i}$ are, the more mutual information between $i$ and the generated image is achieved. In the same way, we define $L_{mi}(G_A, Q_A)$.

The learning of our model consists both to minimize the total loss w.r.t the encoders and generators and to maximize it w.r.t the discriminators :

$$\min_{E_A,E_B,G_A,G_B} \max_{D_A,D_B} \mathcal{L}(E_A, E_B, G_A, G_B, D_A, D_B) = \\ \mathcal{L}_{dis}^{x_A} + \mathcal{L}_{dis}^{x_B} + \lambda_x(\mathcal{L}_{rec}^{x_A} + \mathcal{L}_{rec}^{x_B}) + \lambda_c(\mathcal{L}_{rec}^{c_A} + \mathcal{L}_{rec}^{c_B}) \\ +\lambda_s(\mathcal{L}_{rec}^{s}) - \lambda_{mi}(L_{mi}(G_A, Q_A) + L_{mi}(G_B, Q_B)) \tag{7.10}$$

where $\lambda_x$, $\lambda_c$, $\lambda_s$ and $\lambda_{mi}$ represent the importance of each loss. In our trainings, we set $\lambda_x = 10$, $\lambda_c = \lambda_s = \lambda_{mi} = 1$ as the image reconstruction is the most important loss in our structure.

### 7.2.3 Few-shot learning architecture

We also aim to address the problem of few-shot learning by proposing some extensions in our method. The architecture of our extended work, F-InfoMUNIT, is shown in Figure 7.3. In addition to the InfoMUNIT's components, F-InfoMUNIT adopts a content discriminator $D_c$ (violet) that learns to predict the domain of content codes. Simultaneously, the encoders now have one more objective, which is to fool this discriminator. This discriminator is inspired by the domain-invariant feature space from [190]. Another additional component of F-InfoMUNIT is the content classifier $C_c$ (yellow), which learns to predict the content code label. Because training $C_c$ requires labels, it is fed using only content codes extracted from labeled samples.

## 7.3 Experiments

### 7.3.1 Implementation Details

Our network consists of a content encoder, a style encoder, a generator, and a discriminator for each domain. We give the implementation details of each of these networks.

#### Content Encoder.

Input images are firstly led to the content encoder, where they are down-sampled by strided convolutional layers and further processed by residual blocks. We apply Instance Normalization for all convolutional layers in the content encoder. The output of the content encoder is the content code in the form of a tensor.

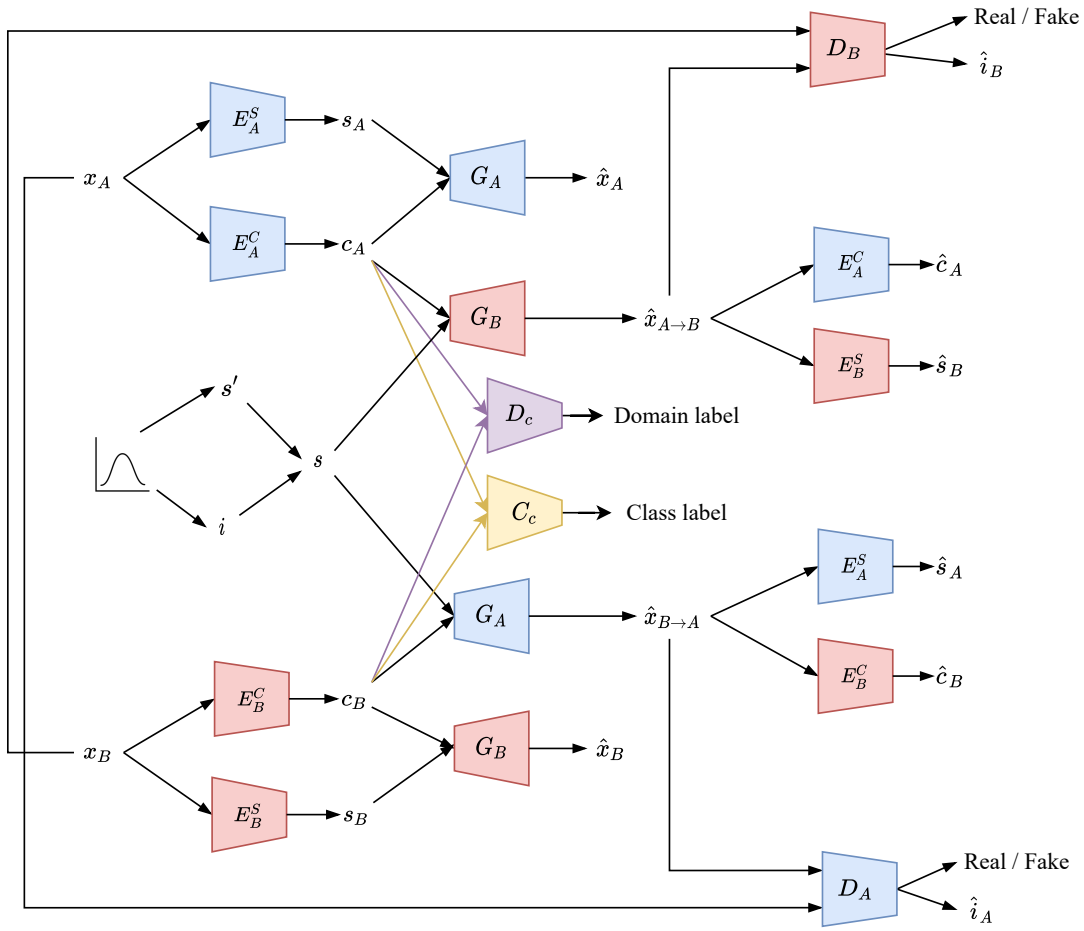



Figure 7.3: Overview of F-InfoMUNIT.

**Style Encoder.**

Similarly, the style encoder also down-samples input images using strided convolutional layers and a global pooling layer. A fully connected (FC) layer is applied to produce a style code as a vector consisting of 8 digits, in which 2 final digits represent the information code (disentangled style) $I_i$ of the image.

**Generator.**

The generator takes content code and style code as inputs to reconstruct the initial input image. The content code goes through residual blocks and upsampling layers. These residual blocks are upgraded with Adaptive Instance Normalization (AdaIn) layers [72] which receive style parameters from a multilayer perception (MLP) which has the style code as its input.

**Multi-purpose Discriminator.**

Our discriminator consists of two branches. The first branch is a traditional discriminator, which can be found in most of the GAN-based models. The second branch consists of convolutional blocks to learn the Q distribution. These two branches share the first convolutional blocks.

**Hyperparameters.**

In all our experiments in the paper, we apply Adam optimizer with $\beta_1$ and $\beta_2$ as 0.5 and 0.999 respectively. The learning rate is initially set to 0.0001, with a weight decay of 0.0001 applied every 100 thousand iterations. Our weight losses are $\lambda_x = 10$, $\lambda_c = \lambda_s = \lambda_{mi} = 1$.

#### Baselines

We compare our proposed method InfoMUNIT with the following unpaired image-to-image translation techniques: CycleGAN[188], MUNIT[73] and DRIT++[97]. Those methods' training procedures are done using official source code and configurations provided by their authors on GitHub.com.

### 7.3.2 Evaluation

We use three performance measures that estimate the quality and the diversity of the generated images to compare InfoMUNIT with the baselines.

#### Conditional Inception Score

Based on Inception Score (IS) [139], Huang et al. [73] introduced Conditional Inception Score (CIS) specified for evaluating multimodal image-to-image tasks. While IS measures the quality and diversity of all generated images at once, CIS focuses on the diversity of images translated from the same input image. Having multiple input images in the test set, we compute CIS for each group of images generated from the same input, and finally, take the mean CIS for the whole test set.

#### Fréchet Inception Distance

Fréchet Inception Distance (FID) [66] computes the distance between the set of generated images and the target domain's set images. It is computed by calculated the distance between the Inception feature vectors for the two sets of images. Thus, FID can be used for evaluating networks that are trained on specific datasets without requiring a classifier pre-trained on a similar dataset. The lower FID we have, the more realistic the generated images are. Normally, those feature vectors are taken from the third pooling layer of the Inception model, which contains 2048 features. Due to the small size of our datasets, we compute the distance using features of the second pooling layer containing 192 features.

#### LPIPS Distance

The translation diversity is also measured by LPIPS distance which is shown in [186] to be highly correlated with human judgment. We compute LPIPS distance, with AlexNet as the backbone network, on generated samples of each input image, then take the average value. The larger distance among them, the more diverse they are.

#### Few-shot classification

To evaluate the performance of our models on the task of few-shot domain adaptation, we first train them to translate images from the source domain to the target domain. Using the generated images, we train a classifier to predict the label of samples from the test dataset from the target domain. As an unsupervised domain adaptation method, InfoMUNIT does not have access to labels of training data. F-InfoMUNIT, additionally, accesses to labels of all training data from the source domain and one labeled sample from each category of the target domain. Other methods such as DANN, FADA, and F-CADA used for the comparison already have their own classifier embedded in their architecture, so there is no need to have an additional classifier for them. Note that DANN, like MUNIT and InfoMUNIT, does not access any labels either, making its training "zero-shot".

### 7.3.3 Datasets

We use multiple datasets for evaluating InfoMUNIT and compare its performance with state-of-the-art techniques on the task of image-to-image translation. Each dataset contains two sets of images, and our network is trained to transform images between the two domains. We crop and down-sample all images to the size of $64 \times 64$, in RGB-color mode.

#### Edges↔Shoes and Edges↔Bags

These two datasets contain images of shoes and handbags along with their edges, introduced in Isola et al. [80]. The edges↔shoes dataset contains 138667 pairs of samples while the edges↔bags dataset contains 49925 pairs. From each dataset, we keep 200 pairs of samples for testing and the rest for training. Note that we do not use the paired information of these two datasets.

#### Cats↔Dogs

The dataset comprises 1364 photos of dogs and 871 photos of cats, cropped to their heads [97]. We keep 100 images from each set for testing while the rest is used for training.

#### Portraits (Painted↔Real)

This dataset consists of 1814 painted portraits and real 6452 portraits captured by cameras [97]. We keep 100 images from each set for testing while using the rest for training.

#### SVHN↔MNIST

For our few-shot domain adaptation experiments, we train F-InfoMUNIT to transform images between two data domains, represented by two datasets MNIST [95] and SVHN [122]. The MNIST dataset contains images of handwritten digits in the form of binary images. Images are at the size of $28 \times 28$, and each one contains one digit. The value of the digit is also the label of the image. The dataset consists of 60000 training images and 10000 testing images. The SVHN dataset is a collection of photos of street view house numbers. Each digit is cropped and labeled by its digit value. In total, the dataset has 73257 digits for training and 26032 for testing. For our experiments, all images are resized to $64 \times 64$ and converted to the RGB format.

## 7.4 Results

### 7.4.1 Image Quality

The qualitative comparison of InfoMUNIT and other methods is shown in Figure 7.4. The objective of InfoMUNIT is to increase the diversity and ability to control features of generated images compared to MUNIT and the state-of-the-art, while not hurting their quality. As shown in Figure 7.4, the quality of images generated by InfoMUNIT is at least as good as the images from other methods. The result is confirmed in Table 7.1 where we apply FID to evaluate the realism of the generated images quantitatively. Even though InfoMUNIT does not out outperform other methods in terms of image quality in any task, its performance is stable across all tasks. The performance of InfoMUNIT is close to the best method for each dataset. InfoMUNIT performs equivalent to or better than MUNIT. This shows that the disentangled features also have an impact on the quality of the images. Notice that DRIT++ is the best for the first four tasks but totally fails in the last four tasks. This is illustrated by the strange dog images generated by DRIT++ in Figure 7.4. On the opposite, CycleGAN gives the best performance for the last four tasks but is bad in the first four tasks and especially


Figure 7.4: Random samples generated by our method and baselines, trained on two datasets: edges→bags (left) and cats→dogs (right). The input images (and ground-truths) are displayed in the first column. Other columns show random outputs of baseline methods and InfoMUNIT.

| | **InfoMUNIT** | **MUNIT** | **CycleGAN** | **DRIT++** |
|---|---|---|---|---|
| edge2bag | 2.81 | 2.56 | 4.23 | **1.69** |
| bag2edge | 7.68 | 8.52 | 58.53 | **5.64** |
| edge2shoe | 1.28 | 1.44 | 4.86 | **1.13** |
| shoe2edge | 4.69 | 8.83 | 88.03 | **4.24** |
| dog2cat | 9.24 | 13.48 | **2.56** | 21.59 |
| cat2dog | 6.31 | 6.31 | **2.02** | 18.14 |
| paint2real | 2.96 | 3.02 | **2.56** | 7.29 |
| real2paint | 8.60 | 8.51 | **3.97** | 18.85 |
| Average | **5.45** | 6.58 | 20.84 | 9.82 |

Table 7.1: Fréchet Inception Distance (FID). Lower value means better performance.

in the bag2edge and shoe2edge tasks. On average, InfoMUNIT achieves the best FID value among the four image-to-image translation methods. The good quality of images generated by InfoMUNIT is stable on multiple datasets.

### 7.4.2 Image Diversity

Table 7.2 and Table 7.3 respectively show the CIS and LPIPS scores that evaluate the diversity of generated images. CycleGAN is not a multimodal method and can generate only one output from one input, so it does not appear in this table. The LPIPS and CIS scores of InfoMUNIT are clearly superior to the scores of DRIT++ and MUNIT. The only exceptions are for the shoe2edge task where the LPIPS of DRIT++ is higher and for the real2paint task where the LPIPS of MUNIT is higher. In both cases, the LPIPS of InfoMUNIT is very close to the best score and still higher than the LPIPS of the third method. Overall datasets, the scores of InfoMUNIT are significantly better than the other methods. Figure 7.5 and Figure 7.6 illustrate the higher diversity of InfoMUNIT compared to MUNIT. These results show that InfoMUNIT generates significantly more diverse outputs than MUNIT and DRIT++.

### 7.4.3 Controlling Features

In this subsection, we show the advantage of InfoMUNIT over its predecessor MUNIT in manipulating features. From Figure 7.5, we can observe that varying values of style code in MUNIT can lead to slight changes like the color of the object. With InfoMUNIT, we can significantly manipulate the features of the object. The first disentangled feature controls the bag's size,

| | **InfoMUNIT** | **MUNIT** | **DRIT++** |
|---|---|---|---|
| edge2bag | **3.00** | 2.07 | 2.13 |
| bag2edge | **2.01** | 1.07 | 1.60 |
| edge2shoe | **2.35** | 2.23 | 1.76 |
| shoe2edge | 1.44 | 1.00 | **1.51** |
| dog2cat | **2.24** | 1.97 | 1.11 |
| cat2dog | **2.65** | 2.24 | 1.09 |
| paint2real | **1.96** | 1.91 | 1.74 |
| real2paint | 2.14 | **2.26** | 1.14 |
| Average | **2.40** | 1.88 | 1.51 |

Table 7.2: LPIPS distance. Higher value means better performance.

| | **InfoMUNIT** | **MUNIT** | **DRIT++** |
|---|---|---|---|
| edge2bag | **0.42** | 0.29 | 0.30 |
| bag2edge | **0.35** | 0.04 | 0.22 |
| edge2shoe | **0.26** | 0.24 | 0.22 |
| shoe2edge | **0.24** | 0.00 | 0.12 |
| dog2cat | **0.32** | **0.32** | 0.04 |
| cat2dog | **0.30** | 0.28 | 0.03 |
| paint2real | **0.25** | **0.25** | 0.11 |
| real2paint | **0.33** | 0.30 | 0.06 |
| Average | **0.31** | 0.21 | 0.14 |

Table 7.3: Conditional Inception Score (CIS). Higher value means better performance.


Figure 7.5: Manipulating two last digits in the style code of MUNIT and InfoMUNIT on edges→bags task.

(a) Input

(b) Varying two last digits of style code of **MUNIT**

(c) Varying two last digits of style code of **InfoMUNIT**

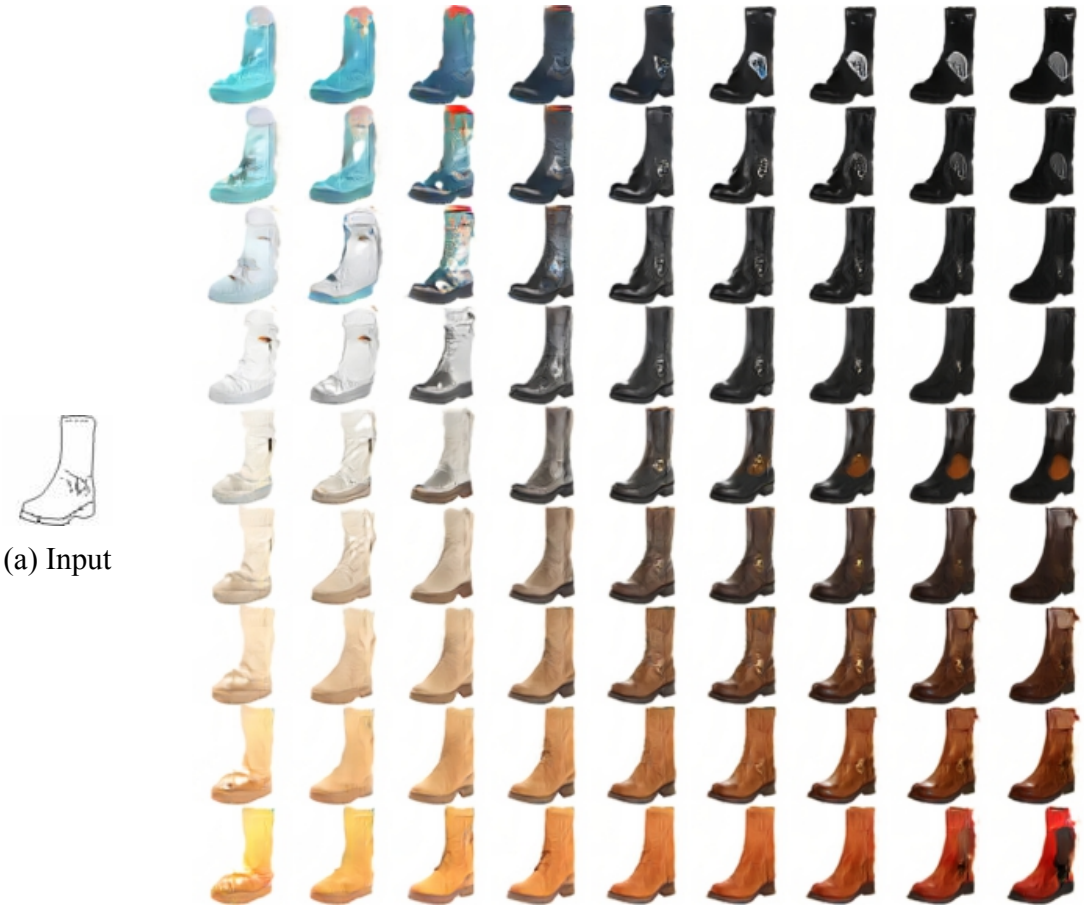
Figure 7.6: Manipulating two last digits in the style code of MUNIT and InfoMUNIT on edges→shoes task.

and the second one controls the color from white to black. We also notice that InfoMUNIT can propose different textures of the bag.

The performance of InfoMUNIT on the edges→shoes task is illustrated in Figure 7.6 and Figure 7.7. While MUNIT can only change some small details of the shoes, we can significantly manipulate the color of the shoes with InfoMUNIT. Varying the first info style code makes the color changed from bright to dark while varying the second one changes the color from cold to warm. In Figure 7.6, we can see that the first info style code is also responsible for the shoes' style. From left to right, it turns a sneaker into a pump and makes it darker at the same time. This effect makes sense as pumps are more likely to have dark colors than sneakers. The second info-style code turns a dark, sporty shoe into a brown leather shoe. In Figure 7.7, we combine the two features represented by the two last two dimensions of the style code of InfoMUNIT. From left to right, we increase the value of the first informational style code, while the value of the second informational style code is increased when traveling from top to bottom. Going down the vertical direction, we see that the shoe color becomes warmer, while following the horizontal direction, the color becomes darker.

Please note that the value of each disentangled feature in this test is plotted from $-2$ to 2 instead of $-1$ to 1 in the training phase, which means the generator is receiving style code values that it has never seen before. This explains why the images on the border look a bit extreme and unrealistic. We plot them out to study the meaning of the learned features instead of evaluating the output realism.

We run the same test on the gingivitis dataset and receive results in Figure 7.8 and Figure 7.9. As shown in Figure 7.8, by modifying the value of the two last digits of the style code (informational style code in the case of InfoMUNIT), both models can make some changes to the output features. MUNIT seems to learn to make slight modifications on the height of teeth and the color of the gum, while InfoMUNIT makes changes on the appearance of the cheek-retractor and the level of infection of the gum. By varying the two last digits of InfoMUNIT, we obtain significantly more changes than in the case of MUNIT. This result is similar to the result that we found in Figure 7.5 and Figure 7.6. In Figure 7.9, we combine the two features learned by InfoMUNIT and stored them into the two last digits of the style code. Moving from left to right and top to bottom, we get similar trends as in Figure 7.8. Moreover, we also observe on the right of the figure that increasing the value of the second feature, which leads to more red gingiva, also slightly reduces the amount of cheek-retractor in the image. This phenomenon can be explained by the fact that gingivitis is more likely to be observed near the front teeth than molars, where the gum is usually partially covered by the retractor. This helps us to understand more about the characters of the dataset.



(a) Input

(b) Outputs generated by **InfoMUNIT**

Figure 7.7: Combination of the two last digits in the style code of InfoMUNIT on edges→shoes task. From left to right (b), we vary the value of the first information latent code. From top to bottom, we vary the second one.

### 7.4.4 The length of information latent code

We perform some experiments to investigate the impact of the length of information latent code $i$ on the generated images varying from 1 to 8. Table 7.4 shows some of these results on the edge2shoe datasets. We see that the FID, CIS and LPIPS weakly vary with the length of $i$. We conclude from these results that the quality and diversity of the generated images by InfoMUNIT are robust to the information latent code's length.

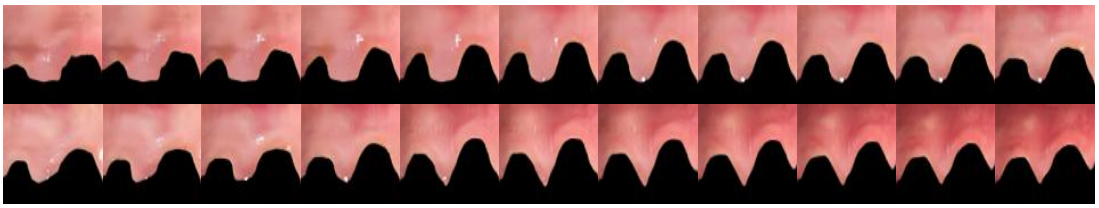
| **Length of i** | **1** | **2** | **4** | **6** | **8** |
|---|---|---|---|---|---|
| **FID** | 2.99 | **2.81** | 3.19 | 3.11 | 3.37 |
| **CIS** | 2.59 | 3.00 | 3.64 | 3.61 | **3.65** |
| **LPIPS** | 0.42 | 0.42 | **0.47** | **0.47** | 0.46 |

Table 7.4: Performance of InfoMUNIT with different lengths of information latent code.

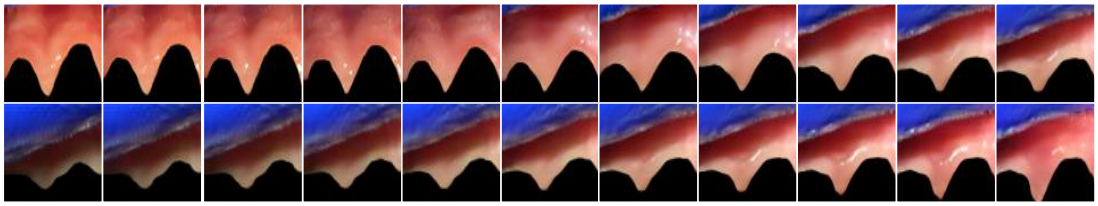
### 7.4.5 Few-shot Domain Adaptation

We extend our method to implement a few-shot domain adaptation and test it by training a classifier on generated samples. Our method F-InfoMUNIT is trained for a one-shot domain adaptation task, and the performance is compared with state-of-the-art few-shot domain adaptation methods in Figure 7.10. Even though InfoMUNIT is not proposed for few-shot learning,

(a) Input



(b) Varying two last digits of style code of **MUNIT**



(c) Varying two last digits of style code of **InfoMUNIT**

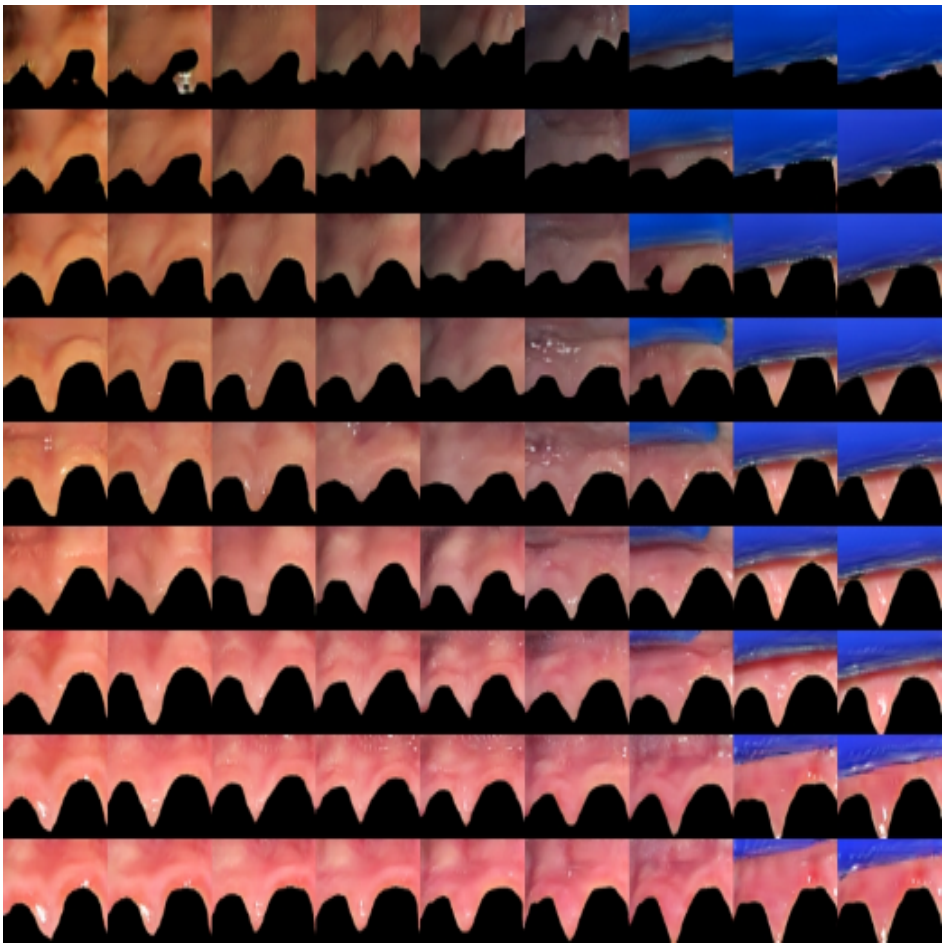
Figure 7.8: Manipulating two last digits in the style code of MUNIT and InfoMUNIT on the task healthy→gingivitis.

it stays on the list to show the improvement of F-InfoMUNIT to its original architecture. As can be seen from the figure, F-InfoMUNIT significantly outperforms DANN and FADA in this task but still achieves 4.79% lower classification accuracy than the state-of-the-art model F-CADA. However, compared to the original InfoMUNIT, it is clear that the additional content classifier and content discriminator in F-InfoMUNIT make a great gain in the classification performance (31.74% accuracy). We believe that our method can be improved even further and achieve higher performance in future works.

## 7.5 Conclusion

We proposed an extension of MUNIT called InfoMUNIT, which can manipulate features of the translated images. Our method is demonstrated in multiple image-to-image translation tasks. It achieves comparable translated image quality to state-of-the-art approaches and outperforms them in terms of output diversity. Moreover, our method improves the user's control of the generated images, this kind of tool can make the image manipulation method more usable for real-life applications. We also extend our method to perform one-shot domain adaptation (F-InfoMUNIT) and achieve promising classification accuracy.

(a) Input



(b) Various outputs generated by InfoMUNIT

Figure 7.9: Combination of the two last digits in the style code of InfoMUNIT on healthy→gingivitis task. From left to right (b), we vary the value of the first information latent code. From top to bottom, we vary the second one.

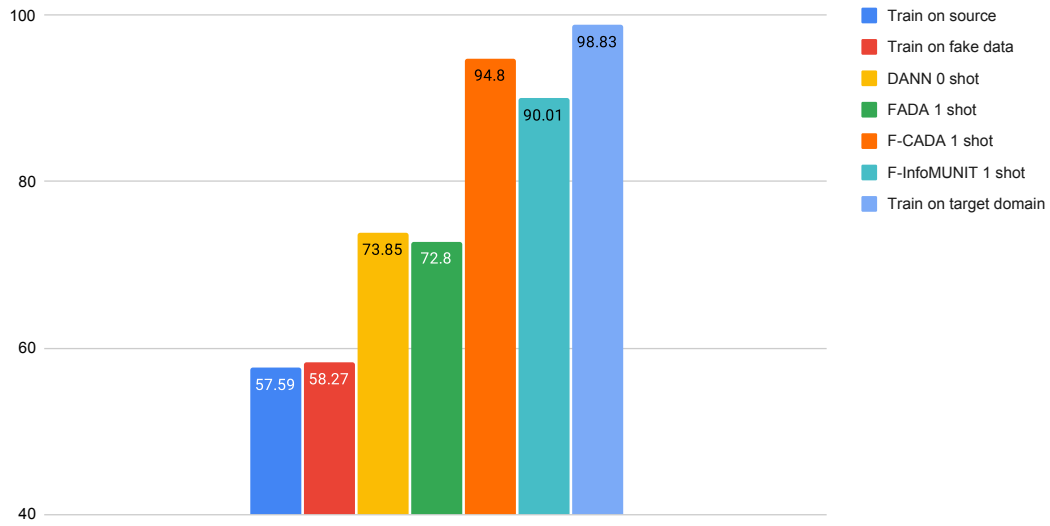



Figure 7.10: Classification accuracy on the test data (target domain) of few-shot learning models including FADA, F-CADA and F-InfoMUNIT on the one-shot learning task DANN, in particular, does not learn from any target samples making it zero-shot. We also show the results when the classifier is trained only on source data (lower bound), only on generated data and directly on the target domain (upper bound).

# Chapter 8

# Conclusion

This chapter summarizes the thesis's contributions, their advantages and drawbacks, experiences learned during our research and some suggestions for future works.

## 8.1 Summary of Contributions

This section summarizes contributions being made during three years of the doctoral program with a short explanation of how they are linked to the template of the thesis.

In Chapter 4, we propose a preprocessing pipe-line that includes pre-trained teeth detection and recognition models and training a deep learning model to detect the presence of gingivitis on images of teeth. The model is evaluated in 10-fold cross-validation and achieves an accuracy of $90.88 \pm 3.25\%$ and an AUC of $95.52 \pm 1.88\%$. We learn that combining the model's prediction on multiple views provides a more accurate final result, which also helps us take advantage of our dataset, where we usually have multiple views of all teeth. Besides the traditional data augmentation technique, we also train an unsupervised image-to-image translation model (MUNIT [73]) that learn to generate synthetic samples of healthy samples and infected samples. Generated samples are mixed with real training data. However, our experiments show that this approach does not improve the classification model even though the generated outputs look realistic for human eyes. From the saliency maps, we find out that our model actually pays a lot of attention to the boundary between the gum and the teeth, which supports the idea that CNNs predictions are based on the texture and the structural information of objects. The diagnosis process of Dental Mind includes the detection of gingivitis on patients' gums. Therefore, this work assists dental technicians at the firm to complete their tasks more quickly. One of the main limitations of this work is the binary classifier which only gives two answers, yes/no, to the presence of gingivitis. However, the formation of gingivitis is a continuous process, including multiple phases. Therefore, the prediction can be even more precise if the classifier can distinguish multiple levels of gingivitis.

In Chapter 5, we apply a supervised image-to-image translation model (Pix2pixHD [173] as a data augmentation method for our CNNs-based dental crowding classifier. Additional training samples are generated by Pix2pixHD, which is trained to generate dental images from projections of 3d models of teeth. Plus, we propose additional augmentations operations in the 3d environment to generate even more samples. As a result, our classifier's accuracy is improved from 71.45% to 82.10% thanks to three thousand synthetic samples. This result demonstrates the advantage of supervised image translation models that can be used when paired data is available over unsupervised ones trained using unpaired data. We also realize that our model's accuracy increases with the number of generated images to a saturation point. Then, if the generated images continue to be added, the model's accuracy will decrease because it is over-fitted with generated data. There is still a gap between the distribution of generated sample and the real dataset, even though they look very realistic for humans. The work can be extended by replacing the binary classifier with one that can distinguish among multiple

levels of dental crowding. Similar to the work on gingivitis, the prediction is limited to a binary classification model. Therefore, training a classifier that categories multiple types of crowding will be more beneficial. Besides, the model Pix2pixHD that is trained in our experiments is still far from perfectly capturing the contribution of the training dataset, causing a drop in the classifier's performance when being trained with a large amount of generated images. Improving the performance of the GAN-based image translation model would improve the quality of generated images, raising the classification model's performance.

In Chapter 6, we adopt two image translation methods CycleGAN[188] and StarGANv2[27], for generating aligners on images of teeth. Our experiments show that both methods succeed in learning to generate realistic images, but StarGANv2 is outstanding in multiple ways. Firstly, on our computation device GTX 1080Ti, CycleGAN is limited to the size of $320 \times 320$ while StarGANv2 can work with images with the size up to $1024 \times 1024$, which is considered high-resolution. Secondly, images generated by StarGANv2 are more realistic and more diverse than CycleGAN's output according to FID and LPIPS metrics. The qualitative comparison also shows that CycleGAN generates images containing more artifacts than ones generated by the other method. Therefore, StarGANv2 is the suitable image translation model for our task. We also propose a pipe-line that takes the images from the orthodontist, preprocess these images, gets them translated by the pre-trained model, recolors them using the original image as reference and sends the result back to the orthodontist. The project directly contributes to the application of the host company. For future work, it would be helpful to study and optimize the model's size and inference time. It would significantly change the user experience if the transformation could be done in real-time.

In Chapter 7, we propose InfoMUNIT, a new unsupervised multimodal image-to-image translation method that learns to disentangle features of the training data by maximizing the mutual information between style latent codes and generated images. The backbone of the model is taken from MUNIT, [73] and the discriminator is modified based on the idea of InfoGAN [26]. We experiment with the model on multiple well-known datasets and compare our model with state-of-the-art unsupervised image-to-image translation methods. InfoMUNIT outperforms others in terms of image quality (FID) and diversity (CIS, LPIPS). Our method also provides some controls on certain features of the dataset that it learns without any annotation. By tweaking the values of the style code, one can modify the characteristics of the generated images. Thanks to our method, we can learn the mapping between data domains and modify the generated images to have desired features. The method is also extended to perform one-shot domain adaptation, called F-InfoMUNIT. The model achieves promising classification accuracy, which is comparable to state-of-the-art works.

## 8.2 Future Work

In this section, we propose some perspectives to improve our work in the future.

As said above, the gingivitis detection model in Chapter 4 can be extended by working with multiple gingivitis levels. We propose two approaches to address the problem. Firstly, the classifier will be modified to classify more than two classes, and the dataset will be relabeled accordingly. For example, if we choose to distinguish between four levels of gingivitis (including one class for healthy samples), the classifier must produce a four-dimension vector representing its prediction, and each sample in the dataset must be tagged with one of the four labels. The second approach is to represent different gingivitis stages on a scale between 0.0 and 1.0 (0 for completely healthy samples and 1.0 for extremely noticeable gingivitis) and train a regression CNN model that predicts the score. The advantage of this idea is that the model learns the training loss related to the levels of gingivitis. For the first approach, the four classes are treated equally. In the second approach, predicting a healthy case as noticeable gingivitis leads to a higher loss than predicting it as slight gingivitis. However, the second approach makes it more complicated for annotation. In short, both directions are worth to be explored in different ways.

It is also interesting to explore other GAN-based techniques to augment the training data for this problem as all the methods that we try in this work do not bring any improvement in the performance. Self-supervised learning can be a promising direction. Another possible idea to improve this work is to train one single model that detects multiple periodical problems because they share the same inputs: images of the gingiva. Such a model can reduce the computation cost and increase the inference speed, hence speeding up the whole monitoring process.

The crowding detection model in Chapter 5 can also be improved similarly to the gingivitis detection model. One can train a classifier will more than two classes or a regression model to predict a score representing the crowding level. Labeling such kind of data is actually simpler in this case than the case of gingivitis because we have access to the 3d models of teeth. Knowing each tooth's position and size, it is straightforward to compute the degree of crowding thanks to existing methods [81]. Data augmentation in the 3d environment can be further explored by slightly applying basic transformations like translation, rotation and scaling on teeth. These modifications will affect the crowding score, so it must be recalculated after the data augmentation process. Another improvement that can be made to this work is improving the performance of the image-to-image translation model used to generate oral images from projections of 3d models. One obstacle when performing data augmentation in the 3d environment is when the generated images are not realistic enough, or more specifically, not close enough to the real dataset's distribution. The more generated images are used for the training, the more saturated the classifier becomes. Thus, having a robust image-to-image translation model is important. As shown in Figure 5.8, our model's classification performance drops when a great number of synthesized images are used in training. Therefore, to benefit from the 3d augmentation operations, it is necessary to improve the translation model or finding another that generates more realistic images.

In Chapter 6, even the quality and the resolution of the generated images are sufficient for our production, there is still room for improvement. Firstly, the generation of output images is a black-box, which means we can not modify the output images' features when necessary. Thus, a study on how to learn disentangled features in a robust image translation method like StarGANv2 would significantly contribute to Dental Mind and the research community. Secondly, the image translation process in Figure 6.1 is always followed by an independent color transferring step, which adds a few seconds to our pipe-line. With a great number of requests being sent to our server every day, these few seconds are significant. In the architecture of StarGANv2, most of the non-domain-specified characteristics are preserved after the transformation, but the image's color is still somehow changed. Making the generator keeping the color condition of the input image could be a great improvement for our application because it removes a post-processing step. One may try to add another training objective that minimizes the difference in terms of colors between the input and the generated image so that the generator learns to preserve the original color of the original image at the same time learning to translate the image to the target domain, which means generating aligners in our application.

Our proposed InfoMUNIT in Chapter 7 shows the ability to learn disentangled features in unsupervised image-to-image translation models using an additional learning objective that maximizes the mutual information between the style code and the output image. The image resolution in our experiments is limited to $64 \times 64$ while most image translation methods now can bear with images of size up to $1024 \times 1024$. Therefore, making InfoMUNIT work with higher resolution images is one of our priorities. Simply adding more layers may get the job done, but the architecture would be weighty. Therefore, developing a more efficient architecture would be necessary. Another idea for extending InfoMUNIT is to improve its ability for disentangling features. The results in Section 7.4.3 show that the method can learn multiple features of the dataset. However, multiple features are still mixed in the same info-style dimension, making it difficult to modify features separately. Solving this problem will be a great step forward for translating images with disentangle features. Another direction is to apply the logic of InfoMUNIT to multi-domain image translation methods like StarGANv2, which can translate images at high resolution and convincing details but still lack the controls on the outputs. The

approach is promising because there is only one generator for all the cross-domain translations, so it will learn the features shared among all domains, while each generator of InfoMUNIT can learn features from only one domain.

# Bibliography


[1] Amjad Almahairi et al. "Augmented cyclegan: Learning many-to-many mappings from unpaired data". In: *arXiv preprint arXiv:1802.10151* (2018).

[2] American Dental Association. "Gingival Recession - Causes and treatment". In: *JADA https://jada.ada.org* 138 (2007). Archived at Wayback Machine; accessed January 2021.

[3] American Dental Association. *Oral Health and Well-Being in the United States*. [Online; Accessed January 18th, 2021]. URL: https://www.ada.org/en/science-research/health-policy-institute/oral-health-and-well-being.

[4] Yoshiko Ariji et al. "Automatic detection and classification of radiolucent lesions in the mandible on panoramic radiographs using a deep learning object detection technique". In: *Oral surgery, oral medicine, oral pathology and oral radiology* 128.4 (2019), pp. 424–430.

[5] Martin Arjovsky, Soumith Chintala, and Léon Bottou. "Wasserstein gan". In: *arXiv preprint arXiv:1701.07875* (2017).

[6] Art Orthodontics. *What are orthodontic brackets and how do they work?* [Online; Accessed January 18th, 2021]. URL: https://www.artorthodontics.com/2019/12/orthodontic-brackets-braces-how-do-they-work/.

[7] Aayush Bansal, Yaser Sheikh, and Deva Ramanan. "Pixelnn: Example-based image synthesis". In: *arXiv preprint arXiv:1708.05349* (2017).

[8] David Barber and Felix V Agakov. "Kernelized infomax clustering". In: *Advances in neural information processing systems*. 2006, pp. 17–24.

[9] David Barber and Felix V Agakov. "The IM algorithm: a variational approach to information maximization". In: *Advances in neural information processing systems*. 2003, None.

[10] Shane Barratt and Rishi Sharma. "A note on the inception score". In: *arXiv preprint arXiv:1801.01973* (2018).

[11] BARRIE DENTIST. *Advantages With Ceramic Braces*. [Online; Accessed January 20th, 2021]. URL: https://barriedentist.ca/big-mouth-blog/braces-and-invisalign-articles/what-are-ceramic-braces.

[12] Sagie Benaim and Lior Wolf. "One-sided unsupervised domain mapping". In: *Advances in neural information processing systems*. 2017, pp. 752–762.

[13] Best Braces Explained. *10 good reasons to choose purple braces*. [Online; accessed December 8th, 2020]. URL: http://bracesexplained.com/wp-content/uploads/2018/12/purple-braces.jpg.

[14] Karsten M Borgwardt et al. "Integrating structured biological data by kernel maximum mean discrepancy". In: *Bioinformatics* 22.14 (2006), e49–e57.

[15] Konstantinos Bousmalis et al. "Unsupervised pixel-level domain adaptation with generative adversarial networks". In: *Proceedings of the IEEE conference on computer vision and pattern recognition*. 2017, pp. 3722–3731.

[16] Braces Haven. *Self-Ligating Braces: All You Need to Know*. [Online; Accessed January 18th, 2021]. URL: https://braceshaven.com/self-ligating-braces/.

[17] Wieland Brendel and Matthias Bethge. "Approximating cnns with bag-of-local-features models works surprisingly well on imagenet". In: *arXiv preprint arXiv:1904.00760* (2019).

[18] Darla Burke and Steve Kim. *Malocclusion of the Teeth*. [Online; Accessed January 18th, 2021]. URL: https://www.healthline.com/health/malocclusion-of-teeth.

[19] Victor Carbune et al. "Fast multi-language LSTM-based online handwriting recognition". In: *International Journal on Document Analysis and Recognition (IJDAR)* (2020), pp. 1–14.

[20] F Casalegno et al. "Caries detection with near-infrared transillumination using deep learning". In: *Journal of dental research* 98.11 (2019), pp. 1227–1233.

[21] Castronova, John and Drut, Oleg. *What Are Invisalign Attachments (Buttons): A Complete Guide*. [Online; accessed December 17th, 2020]. URL: https://diamondbraces.com/invisalign/smartforce-invisalign-attachments/.

[22] Javad Chalipa et al. "Comparison of bond strength of metal and ceramic brackets bonded with conventional and high-power LED light curing units". In: *Journal of dentistry (Tehran, Iran)* 13.6 (2016), p. 423.

[23] Amar Cheema and Rajesh Bagchi. "The effect of goal visualization on goal pursuit: Implications for consumers and managers". In: *Journal of Marketing* 75.2 (2011), pp. 109–123.

[24] Hu Chen et al. "A deep learning approach to automatic teeth detection and numbering based on object detection in dental periapical films". In: *Scientific reports* 9.1 (2019), pp. 1–11.

[25] Qifeng Chen and Vladlen Koltun. "Photographic image synthesis with cascaded refinement networks". In: *Proceedings of the IEEE international conference on computer vision*. 2017, pp. 1511–1520.

[26] Xi Chen et al. "Infogan: Interpretable representation learning by information maximizing generative adversarial nets". In: *Advances in neural information processing systems*. 2016, pp. 2172–2180.

[27] 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*. 2020, pp. 8188–8197.

[28] 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, pp. 8789–8797.

[29] Joseph Paul Cohen, Margaux Luck, and Sina Honari. "Distribution matching losses can hallucinate features in medical image translation". In: *International conference on medical image computing and computer-assisted intervention*. Springer. 2018, pp. 529–536.

[30] Navneet Dalal and Bill Triggs. "Histograms of oriented gradients for human detection". In: *2005 IEEE computer society conference on computer vision and pattern recognition (CVPR'05)*. Vol. 1. IEEE. 2005, pp. 886–893.

[31] Dental Monitoring. *Dos and dontés to scan your smile with Dental Monitoring*. [Online; Accessed January 28th, 2021]. URL: https://dental-monitoring.com/wp-content/uploads/2020/04/Dental-Monitoring_Dos_Donts_for_scanning.pdf.

[32] Dental Monitoring. *Monitoring - Dental Monitoring*. [Online; Accessed January 28th, 2021]. URL: https://dental-monitoring.com/monitoring/.

[33] Dental Monitoring. *Vision - Dental Monitoring*. [Online; Accessed January 28th, 2021]. URL: https://dental-monitoring.com/vision/.

[34] Dental Monitoring. *Welcome to SmileMate Virtual Consultation*. Youtube. 2021. URL: https://youtu.be/ZMM6iBPRR0s?t=63.

[35] Guillaume Desjardins, Aaron Courville, and Yoshua Bengio. "Disentangling factors of variation via generative entangling". In: *arXiv preprint arXiv:1210.5474* (2012).

[36] T Dietrich et al. "Evidence summary: the relationship between oral and cardiovascular disease". In: *British Dental Journal* 222.5 (2017), pp. 381–385.

[37] Chao Dong et al. "Learning a deep convolutional network for image super-resolution". In: *European conference on computer vision*. Springer. 2014, pp. 184–199.

[38] DRosenbach. *Photographic representation of calculus on the lingual of the mandibular anterior teeth*. [Online; accessed December 8th, 2020]. URL: https://commons.wikimedia.org/wiki/File:MandibularAnteriorCalculus.JPG.

[39] Vincent Dumoulin, Jonathon Shlens, and Manjunath Kudlur. "A learned representation for artistic style". In: *arXiv preprint arXiv:1610.07629* (2016).

[40] Enaranjaj. *Aligners not fitting well on one tooth*. [Online; accessed December 8th, 2020]. URL: https://www.reddit.com/r/Invisalign/comments/9z59i0/aligners_not_fitting_well_on_one_tooth/.

[41] Andre Esteva et al. "Dermatologist-level classification of skin cancer with deep neural networks". In: *Nature* 542.7639 (2017), p. 115.

[42] Chelsea Finn, Pieter Abbeel, and Sergey Levine. "Model-agnostic meta-learning for fast adaptation of deep networks". In: *International Conference on Machine Learning*. PMLR. 2017, pp. 1126–1135.

[43] Forbes, David. *Treatment of a Patient with Severe Crowding Needing Expansion*. [Online; accessed December 8th, 2020]. URL: http://www.forbesorthodontics.com/treatment-of-a-patient-with-severe-crowding-needing-expansion-treatment/.

[44] Oral Health Foundation. *Orthodontic treatment*. URL: https://www.dentalhealth.org/orthodontic-treatment. (accessed: 12.1.2020).

[45] Maayan Frid-Adar et al. "GAN-based synthetic medical image augmentation for increased CNN performance in liver lesion classification". In: *Neurocomputing* 321 (2018), pp. 321–331.

[46] Maayan Frid-Adar et al. "Synthetic data augmentation using GAN for improved liver lesion classification". In: *2018 IEEE 15th international symposium on biomedical imaging (ISBI 2018)*. IEEE. 2018, pp. 289–293.

[47] Cheng-Yang Fu et al. "Dssd: Deconvolutional single shot detector". In: *arXiv preprint arXiv:1701.06659* (2017).

[48] Huazhu Fu et al. "Deepvessel: Retinal vessel segmentation via deep learning and conditional random field". In: *MICCAI*. Springer. 2016, pp. 132–139.

[49] Yaroslav Ganin et al. "Domain-adversarial training of neural networks". In: *The journal of machine learning research* 17.1 (2016), pp. 2096–2030.

[50] Leon A Gatys, Alexander S Ecker, and Matthias Bethge. "Image style transfer using convolutional neural networks". In: *Proceedings of the IEEE conference on computer vision and pattern recognition*. 2016, pp. 2414–2423.

[51] Robert Geirhos et al. "ImageNet-trained CNNs are biased towards texture; increasing shape bias improves accuracy and robustness". In: *arXiv preprint arXiv:1811.12231* (2018).

[52] Arnab Ghosh et al. "Multi-agent diverse generative adversarial networks". In: *Proceedings of the IEEE conference on computer vision and pattern recognition*. 2018, pp. 8513–8521.

[53] Ross Girshick. "Fast r-cnn". In: *Proceedings of the IEEE international conference on computer vision*. 2015, pp. 1440–1448.

[54] Ross Girshick et al. "Rich feature hierarchies for accurate object detection and semantic segmentation". In: *Proceedings of the IEEE conference on computer vision and pattern recognition*. 2014, pp. 580–587.

[55] Yoav Goldberg. "Neural network methods for natural language processing". In: *Synthesis lectures on human language technologies* 10.1 (2017), pp. 1–309.

[56] Ian Goodfellow et al. "Generative adversarial nets". In: *Advances in neural information processing systems*. 2014, pp. 2672–2680.

[57] Alex Graves and Jürgen Schmidhuber. "Framewise phoneme classification with bidirectional LSTM and other neural network architectures". In: *Neural networks* 18.5-6 (2005), pp. 602–610.

[58] Nihal Fatma Güler, Elif Derya Übeyli, and Inan Güler. "Recurrent neural networks employing Lyapunov exponents for EEG signals classification". In: *Expert systems with applications* 29.3 (2005), pp. 506–514.

[59] Ishaan Gulrajani et al. "Improved training of wasserstein gans". In: *Advances in neural information processing systems* 30 (2017), pp. 5767–5777.

[60] Mohammad Havaei et al. "Brain tumor segmentation with deep neural networks". In: *Medical image analysis* 35 (2017), pp. 18–31.

[61] Kaiming He et al. "Delving deep into rectifiers: Surpassing human-level performance on imagenet classification". In: *Proceedings of the IEEE international conference on computer vision*. 2015, pp. 1026–1034.

[62] Kaiming He et al. "Identity mappings in deep residual networks". In: *European conference on computer vision*. Springer. 2016, pp. 630–645.

[63] Kaiming He et al. "Mask r-cnn". In: *ICCV*. 2017, pp. 2961–2969.

[64] Kaiming He et al. "Spatial pyramid pooling in deep convolutional networks for visual recognition". In: *IEEE transactions on pattern analysis and machine intelligence* 37.9 (2015), pp. 1904–1916.

[65] Sven Helm and Poul Erik Petersen. "Causal relation between malocclusion and periodontal health". In: *Acta odontologica scandinavica* 47.4 (1989), pp. 223–228.

[66] 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*. 2017, pp. 6626–6637.

[67] Geoffrey E Hinton, Simon Osindero, and Yee-Whye Teh. "A fast learning algorithm for deep belief nets". In: *Neural computation* 18.7 (2006), pp. 1527–1554.

[68] Teruhiko Hiraiwa et al. "A deep-learning artificial intelligence system for assessment of root morphology of the mandibular first molar on panoramic radiography". In: *Dentomaxillofacial Radiology* 48.3 (2019), p. 20180218.

[69] Sepp Hochreiter and Jürgen Schmidhuber. "Long short-term memory". In: *Neural computation* 9.8 (1997), pp. 1735–1780.

[70] Judy Hoffman et al. "Cycada: Cycle-consistent adversarial domain adaptation". In: *arXiv preprint arXiv:1711.03213* (2017).

[71] Jiayuan Huang et al. "Correcting sample selection bias by unlabeled data". In: *Advances in neural information processing systems* 19 (2006), pp. 601–608.

[72] Xun Huang and Serge Belongie. "Arbitrary style transfer in real-time with adaptive instance normalization". In: *Proceedings of the IEEE International Conference on Computer Vision*. 2017, pp. 1501–1510.

[73] Xun Huang et al. "Multimodal unsupervised image-to-image translation". In: *Proceedings of the European Conference on Computer Vision (ECCV)*. 2018, pp. 172–189.

[74] Zeshan Hussain et al. "Differential data augmentation techniques for medical imaging classification tasks". In: *AMIA Annual Symposium Proceedings*. Vol. 2017. American Medical Informatics Association. 2017, p. 979.

[75] Forrest N. Iandola et al. "SqueezeNet: AlexNet-level accuracy with 50x fewer parameters and <0.5MB model size". In: *arXiv:1602.07360* (2016).

[76] Satoshi Iizuka, Edgar Simo-Serra, and Hiroshi Ishikawa. "Let there be color! Joint end-to-end learning of global and local image priors for automatic image colorization with simultaneous classification". In: *ACM Transactions on Graphics (ToG)* 35.4 (2016), pp. 1–11.

[77] Sultan Kairzhanovich Imangaliyev et al. *Multi-view learning and deep learning for heterogeneous biological data to maintain oral health*. Universiteit van Amsterdam, 2016.

[78] Mohammad Moslem Imani et al. "The effect of orthodontic intervention on mental health and body image". In: *Open access Macedonian journal of medical sciences* 6.6 (2018), p. 1132.

[79] Sergey Ioffe and Christian Szegedy. "Batch normalization: Accelerating deep network training by reducing internal covariate shift". In: *International conference on machine learning*. PMLR. 2015, pp. 448–456.

[80] 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.

[81] Amandeep S Johal and Joanna M Battagel. "Dental crowding: a comparison of three methods of assessment". In: *European Journal of orthodontics* 19.5 (1997), pp. 543–551.

[82] M-Carmen Juan et al. "Computer-aided periodontal disease diagnosis using computer vision". In: *Computerized medical imaging and graphics* 23.4 (1999), pp. 209–217.

[83] Eugine Kang. *Long Short-Term Memory (LSTM): Concept*. Sept. 2017. URL: `https://medium.com/@kangeugine/long-short-term-memory-lstm-concept-cb3283934359`.

[84] 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.

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

[86] Taeksoo Kim et al. "Learning to discover cross-domain relations with generative adversarial networks". In: *Proceedings of the 34th International Conference on Machine Learning-Volume 70*. JMLR. org. 2017, pp. 1857–1865.

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

[88] Durk P Kingma et al. "Semi-supervised learning with deep generative models". In: *Advances in neural information processing systems*. 2014, pp. 3581–3589.

[89] Alex Krizhevsky, Geoffrey Hinton, et al. "Learning multiple layers of features from tiny images". In: (2009).

[90] Alex Krizhevsky, Ilya Sutskever, and Geoffrey E Hinton. "Imagenet classification with deep convolutional neural networks". In: *Advances in neural information processing systems* 25 (2012), pp. 1097–1105.

[91] Joachim Krois et al. "Deep learning for the radiographic detection of periodontal bone loss". In: *Scientific reports* 9.1 (2019), pp. 1–6.

[92] Tejas D Kulkarni et al. "Deep convolutional inverse graphics network". In: *Advances in neural information processing systems*. 2015, pp. 2539–2547.

[93] Ira B Lamster et al. "The relationship between oral health and diabetes mellitus". In: *The Journal of the American Dental Association* 139 (2008), 19S–24S.

[94] Yann LeCun, Yoshua Bengio, and Geoffrey Hinton. "Deep learning". In: *nature* 521.7553 (2015), p. 436.

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

[96] Hsin-Ying Lee et al. "Diverse image-to-image translation via disentangled representations". In: *Proceedings of the European conference on computer vision (ECCV)*. 2018, pp. 35–51.

[97] Hsin-Ying Lee et al. "Drit++: Diverse image-to-image translation via disentangled representations". In: *International Journal of Computer Vision* (2020), pp. 1–16.

[98] Jae-Hong Lee et al. "Detection and diagnosis of dental caries using a deep learning-based convolutional neural network algorithm". In: *Journal of dentistry* 77 (2018), pp. 106–111.

[99] Jae-Hong Lee et al. "Diagnosis and prediction of periodontally compromised teeth using a deep learning-based convolutional neural network algorithm". In: *Journal of periodontal & implant science* 48.2 (2018), pp. 114–123.

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

[101] Zachary Lipton, Yu-Xiang Wang, and Alexander Smola. "Detecting and correcting for label shift with black box predictors". In: *International conference on machine learning*. PMLR. 2018, pp. 3122–3130.

[102] Ming-Yu Liu, Thomas Breuel, and Jan Kautz. "Unsupervised image-to-image translation networks". In: *Advances in neural information processing systems*. 2017, pp. 700–708.

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

[104] Yun Liu et al. "Richer convolutional features for edge detection". In: *CVPR, IEEE*. 2017, pp. 3000–3009.

[105] David G Lowe. "Distinctive image features from scale-invariant keypoints". In: *International journal of computer vision* 60.2 (2004), pp. 91–110.

[106] Lars Maaløe et al. "Improving semi-supervised learning with auxiliary deep generative models". In: *NIPS Workshop on Advances in Approximate Bayesian Inference*. 2015.

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

[108] Ibrahim Al-Mahdi, Kathleen Gray, and Reeva Lederman. "Online Medical Consultation: A review of literature and practice". In: *Proceedings of the 8th Australasian Workshop on Health Informatics and Knowledge Management*. 2015, pp. 27–30.

[109] Alireza Makhzani et al. "Adversarial autoencoders". In: *arXiv preprint arXiv:1511.05644* (2015).

[110] Ovais H Malik, Ailbhe McMullin, and David T Waring. "Invisible orthodontics part 1: invisalign". In: *Dental update* 40.3 (2013), pp. 203–215.

[111] Sanidhya Mangal, Rahul Modak, and Poorva Joshi. "LSTM Based Music Generation System". In: *arXiv preprint arXiv:1908.01080* (2019).

[112] Deborah Manger et al. "Evidence summary: the relationship between oral health and pulmonary disease". In: *British dental journal* 222.7 (2017), pp. 527–533.

[113] Xudong Mao et al. "Least squares generative adversarial networks". In: *Proceedings of the IEEE international conference on computer vision*. 2017, pp. 2794–2802.

[114] Yuma Miki et al. "Classification of teeth in cone-beam CT using deep convolutional neural network". In: *Computers in biology and medicine* 80 (2017), pp. 24–29.

[115] Riccardo Miotto et al. "Deep learning for healthcare: review, opportunities and challenges". In: *Briefings in bioinformatics* 19.6 (2018), pp. 1236–1246.

[116] Mehdi Mirza and Simon Osindero. "Conditional generative adversarial nets". In: *arXiv preprint arXiv:1411.1784* (2014).

[117] Saeid Motiian et al. "Few-shot adversarial domain adaptation". In: *arXiv preprint arXiv:1711.02536* (2017).

[118] Tsendsuren Munkhdalai et al. "Rapid adaptation with conditionally shifted neurons". In: *International Conference on Machine Learning*. PMLR. 2018, pp. 3664–3673.

[119] Zak Murez et al. "Image to image translation for domain adaptation". In: *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*. 2018, pp. 4500–4509.

[120] Alison M Murray. "Discontinuation of orthodontic treatment: a study of the contributing factors". In: *British Journal of Orthodontics* 16.1 (1989), pp. 1–7.

[121] Ian Needleman et al. "Impact of oral health on the life quality of periodontal patients". In: *Journal of clinical periodontology* 31.6 (2004), pp. 454–457.

[122] Yuval Netzer et al. "Reading digits in natural images with unsupervised feature learning". In: (2011).

[123] Alex Nichol, Joshua Achiam, and John Schulman. "On first-order meta-learning algorithms". In: *arXiv preprint arXiv:1803.02999* (2018).

[124] NVIDIA. *It's Here: The New GeForce GTX 1080Ti Graphics Card*. URL: `https://www.nvidia.com/en-sg/geforce/products/10series/geforce-gtx-1080-ti`. (accessed: February 20th, 2021).

[125] Orthodontics Australia. *Dental Crowding: Causes and Treatment Options*. [Online; Accessed January 18th, 2021]. URL: `https://orthodonticsaustralia.org.au/dental-crowding-causes-and-treatment-options/`.

[126] Power Chains: Why Some Colored Braces are Connected. [Online; October 15th, 2010]. URL: `http://www.oralanswers.com/power-chains-why-some-colored-braces-are-connected/`.

[127] Alec Radford, Luke Metz, and Soumith Chintala. "Unsupervised representation learning with deep convolutional generative adversarial networks". In: *arXiv preprint arXiv:1511.06434* (2015).

[128] Christoph A Ramseier et al. "Identification of pathogen and host-response markers correlated with periodontal disease". In: *Journal of periodontology* 80.3 (2009), pp. 436–446.

[129] Aman Rana et al. "Automated segmentation of gingival diseases from oral images". In: *HI-POCT*. IEEE. 2017, pp. 144–147.

[130] Antti Rasmus et al. "Semi-supervised learning with ladder networks". In: *Advances in neural information processing systems*. 2015, pp. 3546–3554.

[131] 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.

[132] Scott E Reed et al. "Learning what and where to draw". In: *Advances in neural information processing systems*. 2016, pp. 217–225.

[133] Erik Reinhard et al. "Color transfer between images". In: *IEEE Computer graphics and applications* 21.5 (2001), pp. 34–41.

[134] Shaoqing Ren et al. "Faster r-cnn: Towards real-time object detection with region proposal networks". In: *IEEE transactions on pattern analysis and machine intelligence* 39.6 (2016), pp. 1137–1149.

[135] Richmond Orthodontist. *Emergency Information*. [Online; Accessed January 18th, 2021]. URL: `https://horseyorthodontics.com/orthodontic-emergencies/`.

[136] D Roberts-Harry and J Sandy. "Orthodontics. Part 4: Treatment planning". In: *British dental journal* 195.12 (2003), pp. 683–685.

[137] Rosa, Santa and Rafael, San. *How to treat and prevent Receding gums*. [Online; accessed December 8th, 2020]. URL: `https://www.preferreddentalcaresanrafael.com/wp-content/uploads/2020/02/How-to-treat-and-prevent-Receding-gums-min.jpg`.

[138] Olga Russakovsky et al. "ImageNet Large Scale Visual Recognition Challenge". In: *International Journal of Computer Vision (IJCV)* 115.3 (2015), pp. 211–252. DOI: `10.1007/s11263-015-0816-y`.

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

[140] Adam Santoro et al. "Meta-learning with memory-augmented neural networks". In: *International conference on machine learning*. PMLR. 2016, pp. 1842–1850.

[141] Mike Schuster and Kuldip K Paliwal. "Bidirectional recurrent neural networks". In: *IEEE transactions on Signal Processing* 45.11 (1997), pp. 2673–2681.

[142] Falk Schwendicke et al. "Deep learning for caries lesion detection in near-infrared light transillumination images: A pilot study". In: *Journal of dentistry* 92 (2020), p. 103260.

[143] Fet al Schwendicke, W Samek, and J Krois. "Artificial intelligence in dentistry: chances and challenges". In: *Journal of dental research* 99.7 (2020), pp. 769–774.

[144] Jill Seladi-Schulman and Jennifer Archibald. *The Stages of Tooth Decay: What They Look Like*. [Online; Accessed January 18th, 2021]. URL: `https://www.healthline.com/health/dental-and-oral-health/tooth-decay-stages#stages-of-decay`.

[145] Pierre Sermanet et al. "Overfeat: Integrated recognition, localization and detection using convolutional networks". In: *arXiv preprint arXiv:1312.6229* (2013).

[146] Hoo-Chang Shin et al. "Learning to read chest x-rays: Recurrent neural cascade model for automated image annotation". In: *CVPR, IEEE*. 2016, pp. 2497–2506.

[147] Hoo-Chang Shin et al. "Medical image synthesis for data augmentation and anonymization using generative adversarial networks". In: *SASHIMI 2018*. Springer. 2018, pp. 1–11.

[148] Ashish Shrivastava et al. "Learning from simulated and unsupervised images through adversarial training". In: *Proceedings of the IEEE conference on computer vision and pattern recognition*. 2017, pp. 2107–2116.

[149] Karen Simonyan, Andrea Vedaldi, and Andrew Zisserman. "Deep inside convolutional networks: Visualising image classification models and saliency maps". In: *arXiv preprint arXiv:1312.6034* (2013).

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

[151] Jake Snell, Kevin Swersky, and Richard S Zemel. "Prototypical networks for few-shot learning". In: *arXiv preprint arXiv:1703.05175* (2017).

[152] Concetto Spampinato et al. "Deep learning for automated skeletal bone age assessment in X-ray images". In: *Medical image analysis* 36 (2017), pp. 41–51.

[153] Nitish Srivastava et al. "Dropout: a simple way to prevent neural networks from overfitting". In: *The journal of machine learning research* 15.1 (2014), pp. 1929–1958.

[154] Rebecca Joy Stanborough and Christine Frank. *Lingual Braces: The Upside and Downside of Braces on the Back Side*. [Online; Accessed January 20th, 2021]. URL: `https://www.healthline.com/health/lingual-braces-2`.

[155] Statista Research Department. *Frequence de consultation d'un dentiste ou chirurgien-dentiste en France en 2018*. [Online; Accessed January 18th, 2021]. URL: `https://fr.statista.com/statistiques/912651/frequence-de-consultation-de-dentiste-ou-chirurgien-dentiste-france/`.

[156] Flood Sung et al. "Learning to compare: Relation network for few-shot learning". In: *Proceedings of the IEEE conference on computer vision and pattern recognition*. 2018, pp. 1199–1208.

[157] Dinoj Surendran. *Swiss Roll Dataset*. [Online; published 16 May, 2004]. URL: `http://people.cs.uchicago.edu/~dinoj/manifold/swissroll.html`.

[158] Ilya Sutskever, James Martens, and Geoffrey E Hinton. "Generating text with recurrent neural networks". In: *ICML*. 2011.

[159] 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.

[160] Yaniv Taigman, Adam Polyak, and Lior Wolf. "Unsupervised cross-domain image generation". In: *arXiv preprint arXiv:1611.02200* (2016).

[161] Joshua B Tenenbaum and William T Freeman. "Separating style and content with bilinear models". In: *Neural computation* 12.6 (2000), pp. 1247–1283.

[162] Tulsa Modern Dental. *Enamel, the Hero of your Teeth*. [Online; Accessed March 3rd, 2021]. URL: `https://www.tulsamoderndental.com/blog/2020/3/11/enamel-the-hero-of-your-teeth`.

[163] Dmitry V Tuzoff et al. "Tooth detection and numbering in panoramic radiographs using convolutional neural networks". In: *Dentomaxillofacial Radiology* 48.4 (2019), p. 20180051.

[164] Jasper RR Uijlings et al. "Selective search for object recognition". In: *International journal of computer vision* 104.2 (2013), pp. 154–171.

[165] Dmitry Ulyanov, Andrea Vedaldi, and Victor Lempitsky. "Improved texture networks: Maximizing quality and diversity in feed-forward stylization and texture synthesis". In: *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*. 2017, pp. 6924–6932.

[166] Ross D Uthoff et al. "Point-of-care, smartphone-based, dual-modality, dual-view, oral cancer screening device with neural network classification for low-resource communities". In: *PloS one* 13.12 (2018), e0207493.

[167] Tim Van Erven and Peter Harremos. "Rényi divergence and Kullback-Leibler divergence". In: *IEEE Transactions on Information Theory* 60.7 (2014), pp. 3797–3820.

[168] STEVEN F. VAUGHAN. *Using Orthodontic Work To Improve Your Bite Function.* [Online; Accessed January 18th, 2021]. URL: https://contemporary-dental.com/2018/12/using-orthodontic-work-to-improve-your-bite-function/.

[169] Thirumalaisamy P Velavan and Christian G Meyer. "The COVID-19 epidemic". In: *Tropical medicine & international health* 25.3 (2020), p. 278.

[170] Shankeeth Vinayahalingam et al. "Automated detection of third molars and mandibular nerve by deep learning". In: *Scientific reports* 9.1 (2019), pp. 1–7.

[171] Pascal Vincent et al. "Extracting and composing robust features with denoising autoencoders". In: *Proceedings of the 25th international conference on Machine learning*. 2008, pp. 1096–1103.

[172] Allison W. Walker. *The connection between periodontal disease and orthodontics.* May 2011. URL: https://www.dentistryiq.com/articles/2011/05/ortho-perio-connection.html.

[173] 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.

[174] Kathryn Watson and Christine Frank. *What Is a Crossbite and How Is It Corrected?* [Online; Accessed January 18th, 2021]. URL: https://www.healthline.com/health/crossbite.

[175] Stephanie Watson and Christine Frank. *What Are the Different Types of Teeth Called?* [Online; Accessed January 18th, 2021]. URL: https://www.healthline.com/health/teeth-names.

[176] Whistler Dental. *How Often Should You See Your Dentist?* [Online; Accessed January 18th, 2021]. URL: https://whistlerdental.com/blog/how-often-should-you-see-your-dentist.

[177] Yuxin Wu and Kaiming He. "Group normalization". In: *Proceedings of the European conference on computer vision (ECCV)*. 2018, pp. 3–19.

[178] Junyuan Xie, Linli Xu, and Enhong Chen. "Image denoising and inpainting with deep neural networks". In: *Advances in neural information processing systems*. 2012, pp. 341–349.

[179] Ting Yao et al. "Boosting image captioning with attributes". In: *Proceedings of the IEEE International Conference on Computer Vision*. 2017, pp. 4894–4902.

[180] C Yarbrough, K Nasseh, and M Vujicic. "Key differences in dental care seeking behavior between Medicaid and non Medicaid adults and children". In: *Health Policy Institute Research Brief. American Dental Association [Internet]* (2014).

[181] Zili Yi et al. "Dualgan: Unsupervised dual learning for image-to-image translation". In: *Proceedings of the IEEE international conference on computer vision*. 2017, pp. 2849–2857.

[182] Özal Yildirim. "A novel wavelet sequence based on deep bidirectional LSTM network model for ECG signal classification". In: *Computers in biology and medicine* 96 (2018), pp. 189–202.

[183] Muzzafar Zaman. *What Is Gingivitis.* [Online; November 6th, 2018]. URL: https://medium.com/dr-muzzafar-zaman-dental-advice/what-is-gingivitis-d6c52ef63362.

[184] Han Zhang et al. "Stackgan: Text to photo-realistic image synthesis with stacked generative adversarial networks". In: *Proceedings of the IEEE international conference on computer vision*. 2017, pp. 5907–5915.

[185] 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.

[186] Richard Zhang et al. "The unreasonable effectiveness of deep features as a perceptual metric". In: *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*. 2018, pp. 586–595.

[187] Jun-Yan Zhu et al. "Toward multimodal image-to-image translation". In: *Advances in neural information processing systems*. 2017, pp. 465–476.

[188] Jun-Yan Zhu et al. "Unpaired image-to-image translation using cycle-consistent adversarial networks". In: *Proceedings of the IEEE international conference on computer vision*. 2017, pp. 2223–2232.

[189] Zhenyao Zhu et al. "Multi-view perceptron: a deep model for learning face identity and view representations". In: *Advances in Neural Information Processing Systems*. 2014, pp. 217–225.

[190] Han Zou et al. "Consensus adversarial domain adaptation". In: *Proceedings of the AAAI conference on artificial intelligence*. Vol. 33. 01. 2019, pp. 5997–6004.

# Publications

Even though most of the time of this thesis is spent at the company Dental Mind, and publication is not an obligation, we still try our best to publish our work as scientific papers. Our work in Chapter 4: Detecting Gingivitis in Oral Images Captured by Smartphone Cameras using CNN is ready to be submitted as a scientific paper but cannot be submitted at the moment for its confidentiality with the Dental Mind company. The research in Chapter 7 with the proposed method, namely InfoMUNIT, will be published in the 7th International Conference on Image Processing and Pattern Recognition (IPPR 2021) on April 24 25, 2021, Copenhagen, Denmark. These two papers are included in the next pages. During three years of the doctoral programs, other smaller studies, research and tasks are accomplished but are not included in this thesis for the limitation of time.

# Detecting Gingivitis in Oral Images Captured by Smartphone Cameras using Convolutional Neural Networks

No Author Given

No Institute Given

**Abstract.** Periodontal diseases (PDs) are a major cause of tooth loss among adult men and women. During orthodontic treatment, patients are reported to be more likely to have PD infection than before and after the treatment. Therefore, it is important for orthodontic patients to be frequently diagnosed to detect the early signs of PDs - commonly known as gingivitis - to provide timely dental care. In this paper, we propose a methodology to detect the presence of gingivitis from images of teeth captured by smartphone cameras. Our pipeline includes a preprocessing step of image masking and cropping, then a convolutional neural network that makes predictions for each patient based on a set of images. We also study the effect on the classification performance of enlarging the training dataset with traditional data augmentation methods and domain transfer based on Generative Adversarial Networks (GANs). Experimental results show that our classifier is able to distinguish between healthy and inflamed gingivae with an accuracy of 90.88% and the area under ROC curve (AUC) is 95.52%.

**Keywords:** Image synthesis · Generative adversarial networks · Deep learning · Data augmentation · Gingivitis · Periodontal diseases · Dental care.

## 1 Introduction

Periodontal diseases (PDs) are the inflammatory diseases affecting gingiva and the supporting tissues of teeth and also the primary cause of tooth loss in adults. Research shows that the chance of having PDs increases when a person is under an orthodontic treatment [16]. If not treated promptly, PDs can lead to serious diseases such as diabetes, pneumonia due to inhalation, strokes and cardiovascular disease [12]. In contrast, the early stage of PDs, often known as gingivitis, can be treated much more easily than when it has progressed to severe levels. Therefore, diagnosing gingivitis frequently is the key to prevent PDs.

Thanks to the rapid increase of worldwide smartphone ownership, remote health care applications have become more accessible than ever. With smartphones, patients can take pictures of a body part and send them to doctors or even computer programs for diagnosis [1]. In this paper, we detect gingivitis from images captured by phone cameras using convolutional neural networks (CNNs).

## 2 Related work

For the last decade, deep learning models have achieved groundbreaking advancements in many domains, especially computer vision [9]. In the field of medical imaging, deep learning can be applied to various parts of human body. In [1], CNNs were trained for skin cancer detection on smartphone-captured images. In neuroscience, scientists applied deep learning for detection of brain tumors on magnetic resonance (MR) images [4]. CNNs were also used for retinal vessel segmentation [3] from digital fundus photographs.

Despite numerous research of deep learning in medical images processing, applications in dental diagnostics have not been fully explored. The work of Juan et al. in [7] was one of the earliest attempts using computer vision methods to automatically diagnose periodontal diseases by measuring depth probing. Images were obtained by a special camera fitted together with a dental probe. Rana et al. [13] used a convolutional autoencoder for segmenting inflammation from intraoral fluorescence images. Deep learning was also used on periapical radiographs for diagnosis and prediction of periodontally compromised teeth in [10] . The common drawback of these methods is that they require techniques such as x-rays or fluorescent for obtaining images, which means patients still have to be diagnosed at clinics. In this work, we train a CNN for detection of gingivitis with oral photos taken by smartphone cameras instead of using professional clinical equipment. The simplicity of our method facilitates more frequent medical examinations, so that the disease will be detected and treated promptly. To our knowledge, no study has been done on automatically detecting PDs from this type of image.

The lack of annotated data is a popular problem when applying deep learning to medical imaging. Aside from basic augmentation methods (e.g. flipping, random crop, rotating, ...), data synthesis using adversarial learning has been used recently for enlarging training datasets [14]. In this paper, we use a style-transfer model based on generative adversarial network (GAN) to enlarge the training data and test its impact on the prediction performances.

Our work has several contributions. First, we apply deep learning for gingivitis detection in colored images. Second, we develop a pipeline that predicts the presence of gingivitis from photos captured by smartphones. Third, we use style-transfer GANs for data augmentation on dental images.

## 3 Methodology

In this section, we describe a pipeline including a preprocessing phase being composed of masking and cropping, then a deep neural network to predict the existence of gingivitis in a record.

### 3.1 Preprocessing

We aim to train a CNN from patches containing gum region between pairs of teeth, where gingivitis usually appears. We develop a preprocessing procedure

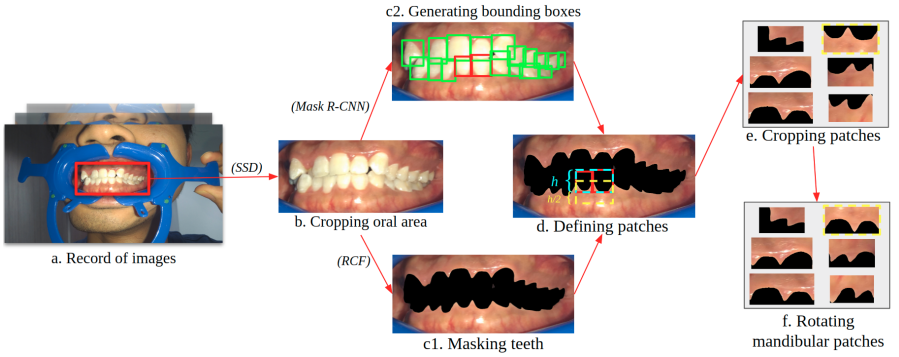



**Fig. 1.** Preprocessing procedure

to extract those patches from the dataset which consists of hundreds of records. Each record contains oral images of a patient from eight poses (Fig. 1a). Images resolutions vary from 611 × 328 up to 3774 × 2261 pixels as they are captured by patients using different devices.

Images from records (Fig. 1a) are cropped to the region of interest (ROI) containing gums and teeth (Fig. 1b). The ROI is defined by Single Shot MultiBox Detector (SSD) [2], a popular object detection algorithm that we fit and use to detect the oral region in images. Since textures of teeth do not contribute to gingivitis detection and might cause the network to learn biases such as braces (because gingivitis often occurs during orthodontic treatments), we mask out the region of teeth in the image using a contour detection and segmentation method namely Richer Convolutional Features (RCF) [11] (Fig. 1c1). Next, Mask R-CNN [5], a deep learning segmentation technique, is applied for teeth detection to obtain bounding boxes of all teeth (Fig. 1c2). We ignore the bounding boxes of molar teeth because they are not usually well-captured. Afterward, as illustrated in Fig. 1d, for each pair of bounding boxes (red), we compute a larger box (cyan) which contains the two boxes. Then, the box is shifted (yellow) toward the gum (upward if the pair of teeth comes from maxillae, and downward if it is from mandibular). The shift distance equals the height $h$ of the large box divided by 2 . Finally, we rotate all mandibular patches by 180 degrees (Fig. 1f) and resize all patches to 128 × 128 (RGB). To avoid deformation, we apply zero-padding to all patches to make them square-shaped before resizing.

### 3.2 Classification model

We train a CNN which classifies those 128 × 128 patches in two classes: *healthy* and *gingivitis*. The network architecture consists of multiple blocks: Convolution + Batch normalization + ReLU, Max-pooling following and Fully connected layers at the end as shown in Fig. 2. Since each pair of teeth is captured from several views, we feed all views of the pair through the CNN and obtain a prediction for each view. The final prediction of a pair of teeth is computed as

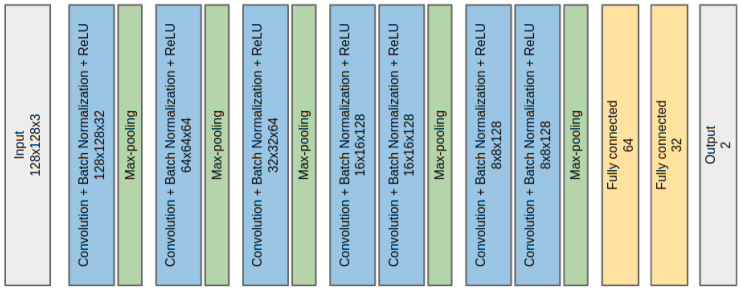



**Fig. 2.** Gingivitis classifier architecture

$p_{final} = \frac{1}{n}\sum_{i=1}^{n} p_i$ where $p_i$ denotes the prediction result of a single view and $n$ is the number of views of the pair.

### 3.3 Data augmentation using style-transfer GAN

Style-transfer networks have been used recently to deal with the problem of lacking data in medical image analysis [14]. In this work, we use a style-transfer model named MUNIT [6] in order to generate *gingivitis* patches from *healthy* patches and vice versa. MUNIT consists of an encoder $E_i$ and a decoder $G_i$ for each data domain $X_i (i = H, G)$ ($H$ represents *healthy* class and $G$ represents *gingivitis* class). The encoder $E_i$ transforms input $x_i$ to become content code $c_i$ and style code $s_i$ as following: $(c_i, s_i) = (E_i^c(x_i), E_i^s(x_i)) = E_i(x_i)$. We implement style transformation by swapping the encoder-decoder pairs. For instance, given a *healthy* patch $x_H$, we can synthesize an artificial *gingivitis* patch $x_{H\to G} = G_G(E_H^c(x_H), s_r)$ in which, $s_r$ is a style latent code drawn from prior distribution $q(s_r) \sim N(0, I)$. Similarly, artificial *healthy* patches can be synthesized as $x_{G\to H} = G_H(E_G^c(x_G), s_r)$ where $x_G$ is a patch from domain *gingivitis*. By modifying $s_r$ we can receive multiple versions of the output which share the same content but differ in style.

We apply MUNIT for data augmentation as described on the left of Fig. 3. First, we train MUNIT with training data of two classes. Then, we apply the pretrained MUNIT for synthesizing artificial data. From each real patch as input, we synthesize three outputs (on the right of Fig. 3) by using three different value of $s_r$. Finally, we combine the original data with artificial data to train our classifier.

## 4 Experiments

### 4.1 Dataset

Images used in this work were collected from (anonymized) database. For this research, we are using 592 records from 280 patients, acquired in six months

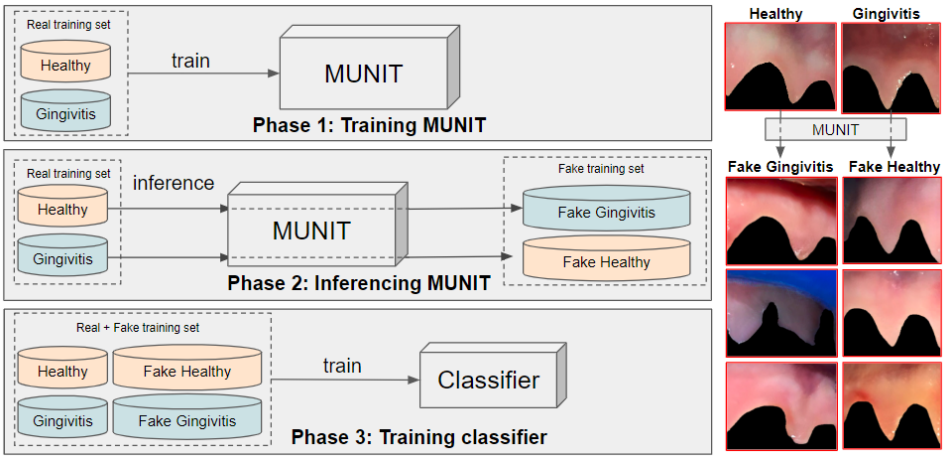



**Fig. 3.** Using MUNIT for augmenting training data to train the gingivitis classifier (left) and some examples of inputs and outputs (right).

in 2018. Each record consists of eight images captured by a smartphone from different poses: left, front and right for opened mouth; left, slightly-left, front, slightly-right and right for closed mouth, with the help of a cheek retractor (Fig. 1a). Photos in our dataset were taken by various types of phones in different lighting conditions. Note that our work has been done locally at (anonymized) and patient information was removed from the dataset. All pairs of teeth in the dataset are labeled by dentists as *healthy* or *gingivitis*. Gingivitis appears in 274 records in the dataset while the other records are completely healthy. We obtain 19978 patches from all records after preprocessing.

### 4.2 Experiments

Our classifier is trained with a batch-size of 128 and Adam optimizer [8] with a learning rate of $10^{-4}$ on 3500 epochs. We validate the model every 15 epochs and compute the average result at the end of the training. We construct two versions of the classifier: one is trained on the given training set and the second one is trained on the same training set mixed with artificial images generated by MUNIT. Using this set of hyperparameters, it takes 50 hours to finish the training on a GeForce GTX 1080 Ti graphic card.

The multimodal style-transfer network is trained using the same hyperparameters as described in the original paper [6] except the number of epochs and batch-size. We train the model for 2800 epochs with batch-size of 14 instead of 1 million iterations and batch-size of 1 for all datasets as in the paper. Input/output size of the model is also reduced from $256 \times 256$ to $128 \times 128$. MUNIT and the classifier share the same training set (Fig. 4). We use the same device that trained our classifier to train MUNIT and it took 9 days for the training.

The accuracy and area under ROC curve (AUC) of the classifiers are estimated by 10-fold cross-validation. Note that MUNIT learning is included in the cross-validation loop in order to avoid that the artificial images overfit the test set. Data is divided at the level of records to assure that images coming from one record will never appear in training set and validation set at the same time. The data of each fold contains about 18,000 patches and 2,000 patches for training and validation respectively.

### 4.3 Results and Discussion

The classifier is evaluated using cross-validation procedure by accuracy and area under ROC curve (AUC). As can be seen from Table 1 and Fig. 4, the classifier achieves better accuracy and AUC by combining the predictions of multiple views. This phenomenon is similar to reality where doctors also need to look at patient teeth from several angles for a proper diagnosing.

The table also shows that adding artificial training data using style-transfer network does not improve classification performance in this case, even though they look convincing in Fig. 3 (on the right side).

We also see that the model performs differently based on the location from where patches were cropped. The model achieved an AUC of $96.17 \pm 2.23$ when evaluated with patches among incisors and canines and an AUC of $94.25 \pm 2.25$ when evaluated with patches among canines and premolars. Since gum regions from the front usually look clearer than ones from the side and the back, they are easier to be recognized by the classifier.

**Table 1.** Accuracy (%) and AUC (%) of classifiers trained with original data and on enlarged data with two prediction techniques: single view and multiple views.

| | Single view | | Multiple views | |
|---|---|---|---|---|
| | Accuracy | AUC | Accuracy | AUC |
| Real data | 88.81 ± 3.55 | 94.98 ± 2.82 | **90.88 ± 3.25** | **95.52 ± 1.88** |
| Real + Artificial data | 88.15 ± 4.07 | 94.81 ± 2.91 | 90.43 ± 3.64 | 95.51 ± 2.00 |

Even if we obtained a high accuracy of gingivitis prediction, we have to check if our classifier has really learnt the concept of gingivitis or based its decision on artifact in the training set. A common approach to interprete the decision of a CNN is to visualize the saliency maps. Saliency maps compute the gradient of the output prediction with respect to the input image and highlight image pixels that contribute the most to the predicted output [15].

As can be seen on the right of Fig. 4, there is a high distribution of white dots on the boundary between gum and (masked) teeth. The density of dots on the gum region is also high but not as much at the boundary. This result suggests that both shape and texture contribute to the classification result, but the shape of the border between teeth and gum seems to be the most important component. The cause of this result may come from different lighting conditions

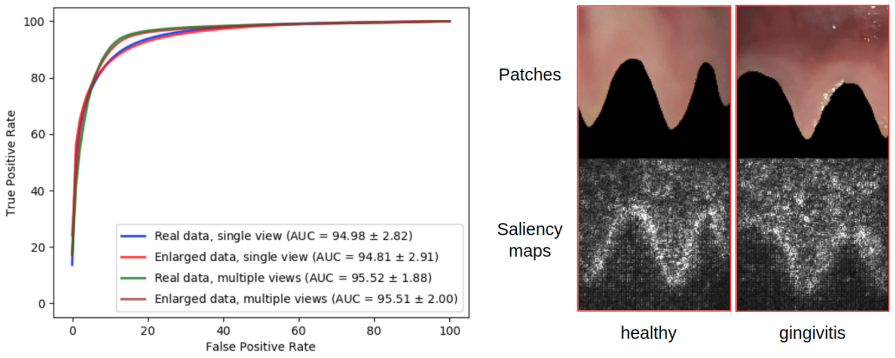



**Fig. 4.** ROC curves and AUC of the classifier when being trained on real data versus enlarged data, evaluated using single view and multiple views (left) and examples of saliency maps generated from patches (right).

between photos in the dataset. The color of the gums can be displayed more boldly in one camera, but lighter in other cameras, while the shape of the gums does not change even if images are taken by different devices and under various conditions. This is coherent to the conclusion of a orthodontist in our team, stating that the most recognizable sign of gingivitis is swollen gum.

## 5 Conclusion

In conclusion, this paper proposes a pipeline to detect gingivitis from photos taken with smartphone cameras with high accuracy. Our approach opens new ways for the diagnosis and follow-up of PDs. We can imagine the following use-case : Initially, a patient wears a cheek retractor and takes images of his/her mouth from different angles. Then, images are sent to a clinic to be automatically preprocessed and given to the classifier to detect gingivitis. Finally, the results are sent back to the patient. Moreover if gingivitis is detected in the images, the system sends the images and results to the dentist and triggers an appointment if needed.

For future work, some questions will be addressed. We will investigate why enlarging the training dataset with style-transfer network does not improve the classification performance, although the artificial images look very similar to real images. We will also improve the model performance by adding preprocessing steps to emphasize the structural information in the images.

## Acknowledgement

We thank (anonymized) for allowing us to work with their dataset and providing computation resources. We are also thankful for (anonymized) for sharing helpful knowledge and tools.

## References


1. Esteva, A., Kuprel, B., Novoa, R.A., Ko, J., Swetter, S.M., Blau, H.M., Thrun, S.: Dermatologist-level classification of skin cancer with deep neural networks. Nature **542**(7639), 115 (2017)
2. Fu, C.Y., Liu, W., Ranga, A., Tyagi, A., Berg, A.C.: Dssd: Deconvolutional single shot detector. arXiv preprint arXiv:1701.06659 (2017)
3. Fu, H., Xu, Y., Lin, S., Wong, D.W.K., Liu, J.: Deepvessel: Retinal vessel segmentation via deep learning and conditional random field. In: MICCAI. pp. 132–139. Springer (2016)
4. Havaei, M., Davy, A., Warde-Farley, D., Biard, A., Courville, A., Bengio, Y., Pal, C., Jodoin, P.M., Larochelle, H.: Brain tumor segmentation with deep neural networks. Medical image analysis **35**, 18–31 (2017)
5. He, K., Gkioxari, G., Dollár, P., Girshick, R.: Mask r-cnn. In: ICCV. pp. 2961–2969 (2017)
6. Huang, X., Liu, M.Y., Belongie, S., Kautz, J.: Multimodal unsupervised image-to-image translation. In: ECCV. pp. 172–189 (2018)
7. Juan, M.C., Alcañiz, M., Monserrat, C., Grau, V., Knoll, C.: Computer-aided periodontal disease diagnosis using computer vision. Computerized medical imaging and graphics **23**(4), 209–217 (1999)
8. Kingma, D.P., Ba, J.: Adam: A method for stochastic optimization. arXiv preprint arXiv:1412.6980 (2014)
9. LeCun, Y., Bengio, Y., Hinton, G.: Deep learning. nature **521**(7553), 436 (2015)
10. Lee, J.H., Kim, D.h., Jeong, S.N., Choi, S.H.: Diagnosis and prediction of periodontally compromised teeth using a deep learning-based convolutional neural network algorithm. Journal of periodontal & implant science **48**(2), 114–123 (2018)
11. Liu, Y., Cheng, M.M., Hu, X., Wang, K., Bai, X.: Richer convolutional features for edge detection. In: CVPR, IEEE. pp. 3000–3009 (2017)
12. Ramseier, C.A., Kinney, J.S., Herr, A.E., Braun, T., Sugai, J.V., Shelburne, C.A., Rayburn, L.A., Tran, H.M., Singh, A.K., Giannobile, W.V.: Identification of pathogen and host-response markers correlated with periodontal disease. Journal of periodontology **80**(3), 436–446 (2009)
13. Rana, A., Yauney, G., Wong, L.C., Gupta, O., Muftu, A., Shah, P.: Automated segmentation of gingival diseases from oral images. In: HI-POCT. pp. 144–147. IEEE (2017)
14. Shin, H.C., Tenenholtz, N.A., Rogers, J.K., Schwarz, C.G., Senjem, M.L., Gunter, J.L., Andriole, K.P., Michalski, M.: Medical image synthesis for data augmentation and anonymization using generative adversarial networks. In: SASHIMI 2018. pp. 1–11. Springer (2018)
15. Simonyan, K., Vedaldi, A., Zisserman, A.: Deep inside convolutional networks: Visualising image classification models and saliency maps. arXiv preprint arXiv:1312.6034 (2013)
16. Walker, A.W.: The connection between periodontal disease and orthodontics (May 2011), https://www.dentistryiq.com/articles/2011/05/ortho-perio-connection.html

# Informative Multimodal Unsupervised Image-to-Image Translation

Tien Tai Doan[1,2], Guillaume Ghyselinck[1], and Blaise Hanczar[2]

[1]Dental Monitoring, Paris, France
t.doan@dental-monitoring.com, g.ghyselinck@dental-monitoring.com
[2]IBISC Laboratory, University of Évry Val d'Essonne, Évry-Courcouronnes, France
blaise.hanczar@ibisc.univ-evry.fr

***Abstract***

*We propose a new method of multimodal image translation, called InfoMUNIT, which is an extension of the state-of-the-art method MUNIT. Our method allows controlling the style of the generated images and improves their quality and diversity. It learns to maximize the mutual information between a subset of style code and the distribution of the output images. Experiments show that our model cannot only translate one image from the source domain to multiple images in the target domain but also explore and manipulate features of the outputs without annotation. Furthermore, it achieves a superior diversity and a competitive image quality to state-of-the-art methods in multiple image translation tasks.*

***Keywords***

*Multimodal Image-to-Image Translation, Mutual Information, GANs, Manipulating Features, Disentangled Representation*

## 1. Introduction

Image-to-image translation can be described as the general problem of mapping an image from one domain to another domain. This seemingly simple approach is the foundation of many applications in the field of computer vision such as colorization [1], style transfer [2], super-resolution [3], denoising, inpainting [4]. Moreover, image-to-image translation has been also applied for data augmentation and achieved competitive results [5] [6] [7]. Based on the availability of data, the problem can be considered as supervised learning where the dataset contains paired samples; or unsupervised learning where the dataset consists of two independent sets of images. This work focuses on the unsupervised image-to-image problem which is more applicable due to its ease of obtaining data but also more challenged in terms of training.

Unsupervised image-to-image translation leads us to the idea that an image in a domain can be translated into multiple images in the second domain, which means the translation can be multimodal. For example, in image colorization, one image can be colored in multiple ways. Some methods [8] [9] have been proposed to use a noisy vector as an additional input of the decoder. The style of the generated images can then be manipulated by changing the values of the style-vector. However, the style-vectors in existing methods are entangled and the translated images are not interpretable as a result. Lacking control over features of the output can be problematic when important information are linked to these features. In the work of Cohen et al. [10], it is shown that CycleGAN was adding/removing tumors from images when transforming MRI images from Flair to T1, especially when there is an imbalance among classes in the training data. Therefore, learning to control the features of the translated images is essential. In this paper, we

propose some improvements on MUNIT [9] - a standard in the field of multimodal image translation - by applying the mutual information maximization technique. Our method, called InfoMUNIT, generates more diverse images and especially can manipulate their textural and structural features without requiring any annotation.

## 2. Related Works

### 2.1. Multimodal unsupervised image-to-image translation

The translation of images from one domain to another has been a challenging problem in computer vision. Thanks to the evolution of convolutional neural networks, especially generative adversarial networks (GANs) [11], many deep learning models have been recently proposed to address the problem of image translation and achieved impressive outcomes.

The research of Isola et al. in [12] is one of the earliest works on image-to-image translation based on GANs. In [13], the method is upgraded using multi-scale generators and discriminators to translate high-resolution images. These methods require paired data for training which is not usually available in practice.

Learning to translate images using unpaired data is more challenging than with paired data because we do not know exactly which data-point in the source domain corresponds to which one in the target domain. Thus, it is reasonable to add some constraints to the training when it is possible. One popular assumption in most image-to-image translation research is that the structure of an image must not be changed too much by the translation. This is similar to language translation, in which, a phrase must have the same meaning after being translated to another language. Shrivastava et al. [14] propose a training strategy in which, a deep network learns to transform the style of synthesized images to make them look more real. To preserve the annotation, they add a pixel-wise loss between the input and output of the style transfer network. Similar approaches are applied in later works such as specific-task loss [15], semantic features [16], or distance between pairs of input samples [17] and so on. These constraints are useful for some specific tasks and datasets but cannot be applied robustly.

Cycle consistency is another well-known loss function being used in many bi-direction image translation models such as DualGAN [18], CycleGAN [19], and DiscoGAN [20]. In these networks, an image being translated from domain A to domain B can be also translated backward to obtain the original image. As this cycle loss is not domain related, it can be applied to most of the bi-direction translation models. In [21], Almahairi et al. extend CycleGAN for learning a many-to-many mapping by combining images with noises. Despite its ease of use, cycle loss does not assure any consistency in terms of annotation which means labels of images can be flipped by the translation. Hoffman et al. [7] proposed to use both cycle consistency and semantic consistency during the training. However, this semantic constraint is not always accessible because it requires a pretrained classifier of a similar dataset.

Another way to preserve the structural information after the transformation is to define a shared latent space where domain-independent features are stored. In UNIT [6], Liu et al. propose to break the translation into two stages: encoding the source image to a latent code and then decoding this code to an image in the target domain. To gain some control over features of the translated image, Huang et al. develop MUNIT as an extension of UNIT, by splitting the latent code into two parts: content and style. With this network, multimodal translation can be done by combining a content code of an image with randomized style codes. The output images inherit content (or structure) from the input image but differ in style (Eg. textures or colors). In DRIT++ [22], a similar idea to

MUNIT is introduced but differs slightly in style transformation techniques. Both MUNIT and DRIT++ store image style in a completely entangled manner, they offer no control over the style of output images despite their diversity. In this work, we extend MUNIT by disentangling the style code without requiring any additional annotations or pretrained networks.

### 2.2. Unsupervised disentangled representation learning

Learning the features of images in an unsupervised fashion has received attention from the computer vision community for years.

Most methods in the early stage were based on restricted Boltzmann machines [23] and stacked auto-encoders [24]. Models in [25] and [26] were proposed for semi-supervised learning and achieved promising results on the MNIST dataset. In [27], a GANs-based method were shown to represent the dataset in a code space where basic linear structures are supported.

Another branch of research uses labeled data to learn disentangled representation. The representation is divided into two parts: one for the given labels and on for other features. Similar fashions of training can be found in different model structures such as bilinear models [28], multi-view perceptron [29], variational autoencoders (VAEs) [30] and adversarial autoencoder [31].

For minimizing the dependency on variation labels, weakly supervised methods were developed. Reed et al. [32] propose correspondence-based training strategies for a higher-order Boltzmann machine consisting of hidden units groups and each group represent a factor of variation. A similar technique is applied to VAE in [33] to manipulate brightness and pose in images of 3D objects. These two methods share one drawback that they require grouped data points which are difficult to collect in real-life applications.

There are not many works on completely unsupervised disentangled representation learning. In [34], hossRBM is introduced as a generalized version of spike-and-slab restricted Boltzmann machine, which entangles variation factors using its higher-order interactions on latent variables. However, the method is not effective in terms of computation cost.

In InfoGAN [35], Chen et al. develop an extension of GAN which maximizes the mutual information between certain variables in the latent code and samples from an unlabeled dataset. This technique enables the model to learn the disentangled representation of images without asking for labels. In this work, we upgrade MUNIT with the mutual information learning objective from InfoGAN to enable it to manipulate features of the translated images.

## 3. Method

Our objective is to translate images from a source domain A to a target domain B, and at the same time to learn the representation of the target domain. Following the idea called *partially shared latent space* in [9], we assume that each image can be encoded as a content code which contains general structural information and a style code which defines how the image will look like. In the state-of-the-art methods, this style latent code is entangled. In this work, we disentangle this style code by maximizing the mutual information between this code and the generated image.

### 3.1. Network architecture

Let $x_A$ and $x_B$ be two images from domain $A$ and $B$ respectively. Our objective is to learn a function $F_{A\to B}$ that projects images from domain $A$ to domain $B$, $\hat{x}_{A\to B} =$

$F_{A\to B}(x_A)$. This function can be decomposed into parts: the encoder and the generator. The encoder $E_A^c$ extracts the content code $c_A$ from the image. The content code is a matrix representing the content of an image independently of its style. The generator $G_B$ generates images in domain $B$ from an content code and a style code $s$ : $\hat{x}_{A\to B} = G_B(c_A, s) = G_B(E_A^c(x_A), [s', i])$. Since we want a one-to-many projection, a style code $s_B$ is inputted in the generator to introduce variability in the generated images. The style code $s$ is a vector created by concatenating two parts $s'$ and $i$ where $s'$ stores entangled style of the generated images, $i$ contains disentangled features of the generated images. $s'$ and $i$ are drawn from a normal distribution $N(0, I)$. The generator learns a function that links the points from a Gaussian distribution to the different ways to apply the style of domain B to a content code. In the same way, we define the function that projects images from domain $B$ to domain $A$ with generator $G_A$ and encoder $E_A^c$. Notice that the content space and style space are common to both domains. This is the generators that project a couple of points from these common spaces to the image sub-spaces corresponding to their domain.

For the learning of these functions, we need to complete our architecture with autoencoders and discriminators. Autoencoders are used to reconstruct the original images from their decomposition into a content code and a style code. Let $E_A^s$ (resp. $E_B^s$) denote the encoder that extracts from an image of domain $A$ (resp. $B$) its style code $s_A = [s'_A, i_A]$ (resp. $s_B = [s'_B, i_B]$). The autoencoder of domain $A$ is therefore defined by $\hat{x}_A = G_A(E_A^c(x_A), E_A^s(x_A))$. Autoencoders are also used to reconstruct the content $\hat{c}_A = E_A^c(G_B(c_A, s))$ and style codes $\hat{s} = E_B^s(G_B(c_A, s))$. The discriminator $D_B$ is used to align the distribution of images produced by the generator $G_B$ with the distribution of original images from domain $A$. It is also used to disentangle the style variables contained in the vector $i$. In the same way, we define the autoencoders $\hat{x}_B = G_B(E_B^c(x_B), E_B^s(x_B))$, $\hat{c}_B = E_B^c(G_A(c_B, s))$, $\hat{s} = E_A^s(G_A(c_B, s))$ and discriminator $D_A$. Figure 1 shows the complete architecture of InfoMUNIT.

### 3.2. Model learning

The training of our model consists to minimize a combination of reconstruction losses and adversarial losses while maximizing the variational mutual information.

Similar to most auto-encoder based architecture, the encoders $E_A^C$ and $E_A^S$ compress input images to content code and style code while the generator $G_A$ takes them to reconstruct the original image from domain $A$. The image reconstruction loss $\mathcal{L}_{rec}^{x_A}$ makes sure the encoder and decoder inverse each other. $L_1$ loss is chosen for the image reconstruction as it usually obtains well the sharpness of the reconstructed image. For the same reason, we have similar reconstruction losses for content code $\mathcal{L}_{rec}^{c_A}$ and style code $\mathcal{L}_{rec}^{s_A}$.

$$\mathcal{L}_{rec}^{x_A} = E_{x_A\sim p(x_A)}[\| G_A(E_A^c(x_A), E_A^s(x_A)) - x_A \|_1] \tag{1}$$

$$\mathcal{L}_{rec}^{x_B} = E_{x_B\sim p(x_B)}[\| G_B(E_B^c(x_B), E_B^s(x_B)) - x_B \|_1] \tag{2}$$

$$\mathcal{L}_{rec}^{c_A} = E_{c_A\sim p(c_A), s\sim\ p(s)}[\| E_B^c(G_B(c_A), s) - c_A \|_1] \tag{3}$$

$$\mathcal{L}_{rec}^{c_B} = E_{c_A\sim p(c_B), s\sim p(s)}[\| E_A^c(G_A(c_B), s) - c_B \|_1] \tag{4}$$

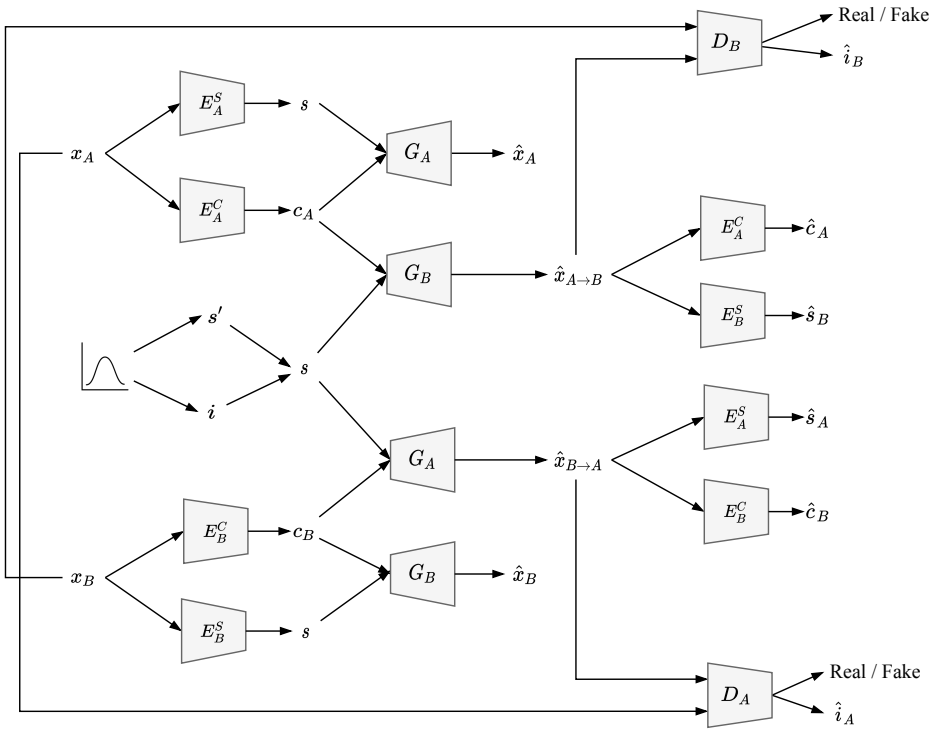



Figure 1: Overview of InfoMUNIT. In this structure, each image is encoded by two encoders into a style code and a content code, and reconstructed by a decoder (also called generator). For translating an image from a domain to another domain, we firstly extract its content code, then combine it with a random style code, and send them both to the generator of the target domain. A part of the style code is used to store disentangled features of output images. We also train a pair of discriminators to distinguish between generated images and real images for each domain. The generators are also trained to maximize the mutual information between features being extracted by those discriminators and the disentangled part in the style code.

$$\begin{aligned}\mathcal{L}^s_{rec} = & E_{c_A\sim p(c_A),s\sim p(s)}[\| E^s_B(G_B(c_A),s) - s \|_1] \\ & + E_{c_B\sim p(c_B),s\sim p(s)}[\| E^s_A(G_A(c_B),s) - s \|_1] \end{aligned} \tag{5}$$

where $p(x_A)$ (resp. $p(x_B)$) is the distribution of images from domain $A$ (resp. $B$), $p(c_A)$ (resp. $p(c_B)$) is the distribution of content code extracted from images from domain $A$ (resp. $B$), and $p(s)$ is the distribution of style code that is the unit Gaussian distribution $N(0, I)$. Note that the distributions $p(c_A)$ and $p(c_B)$ are unknown and the learning set do not contains examples of $c_A$ and $c_B$., we need there fore to generate $c_A$ and $c_B$ samples from the encoders and training images $c_A = (E^c_A(x_A))$ and $c_B = (E^c_B(x_B))$.

The objective of the adversarial losses associated to the discriminators is to align the distributions of the real images with the distribution of the generated images. Like in the GAN, the discriminators try to predict if an image is a real one or an artificial image produced the generator. When the generators are frozen, the generators try to fool the discriminators in generating images close to the real ones. The adversarial losses are

defined by:

$$\mathcal{L}_{adv}^{A} = E_{x_B \sim p(x_B), s \sim p(s)}[log(1 - D_A(G_A(E_B^C(x_B); s)))] + E_{x_A \sim p(x_A)}[logD_A(x_A)] \quad (6)$$

$$\mathcal{L}_{adv}^{B} = E_{x_A \sim p(x_A), s \sim p(s)}[log(1 - D_B(G_B(E_A^C(x_A); s)))] + E_{x_B \sim p(x_B)}[logD_B(x_B)] \quad (7)$$

where the output of the discriminator $D_A(x)$ (resp. $D_B(x)$ ) is the probability that the image $x$ is a real image from the domain $A$ (resp. $B$).

Inspired by the idea of InfoGAN [35], we want a part of the style code to be disentangled features of the output in order to control and improve the diversity of the translated images. The style code is split into two parts $s = [s', i]$. To encourage the subvector $i$ to represent disentangled features of the output, we maximize the mutual information between $i$ and the generated images.

$$\mathcal{I}(i, G_B(c_A, [s, i])) \quad \text{and} \quad \mathcal{I}(i, G_A(c_B, [s, i])) \quad (8)$$

In practice, maximizing this mutual information is not achievable without access to the distribution $P(i|x)$ which is not available in our case. However, according to [36], we can define an additional distribution $Q(i|x)$ as an approximation of $P(i|x)$, and get a lower bound of the mutual information term. Thus we have:

$$\mathcal{I}(i, G_B(c_A, [s, i])) \geq L_{mi}(G_B, Q_B) = E_{i \sim p(i), x_{A \to B} \sim P(G_B(c_A, [s', i]))}[logQ_B(i|x_{A \to B})] \quad (9)$$

Where $p(i)$ is a normal distribution and $P(G_B(c_A, [s', i]))$ is the distribution of the images generated by $G_B$ with the style vector $[s', i]$. In practice, $Q_B$ shares the same layers of the discriminator $D_B$ as they both extract features from $G_B(c_A, [s', i])$ . $Q_B$ is implemented as a secondary output of the discriminator $D_B$ that is notes $\hat{i}$. This means the closer the vector $i$ and predicted vector $\hat{i}$ are, the more mutual information between $i$ and the generated image is achieved. In the same way, we define $L_{mi}(G_A, Q_A)$.

The learning of our model consists both to minimize the total loss w.r.t the encoders and generators and to maximize it w.r.t the discriminators :

$$\min_{E_A, E_B, G_A, G_B} \max_{D_A, D_B} \mathcal{L}(E_A, E_B, G_A, G_B, D_A, D_B) = \mathcal{L}_{dis}^{x_A} + \mathcal{L}_{dis}^{x_B} + \lambda_x(\mathcal{L}_{rec}^{x_A} + \mathcal{L}_{rec}^{x_B}) + \lambda_c(\mathcal{L}_{rec}^{c_A} + \mathcal{L}_{rec}^{c_B}) + \lambda_s(\mathcal{L}_{rec}^{s}) - \lambda_{mi}(L_{mi}(G_A, Q_A) + L_{mi}(G_B, Q_B)) \quad (10)$$

where $\lambda_x$, $\lambda_c$, $\lambda_s$ and $\lambda_{mi}$ represent the importance of each loss. In our trainings, we set $\lambda_x = 10$, $\lambda_c = \lambda_s = \lambda_{mi} = 1$ as the image reconstruction is the most important loss in our structure.

# 4. Experiments

## 4.1. Implementation Details

Our network consists of a content encoder, a style encoder, a generator, and a discriminator for each domain. We give the implementation details of each of these network.

#### 4.1.1. Content Encoder

Input images are firstly led to the content encoder where they are down-sampled by strided convolutional layers and further processed by residual blocks. We apply Instance Normalization for all convolutional layers in the content encoder. The output of the content encoder is the content code in a form of a tensor.

#### 4.1.2. Style Encoder

Similarly, the style encoder also down-samples input images using strided convolutional layers and a global pooling layer. A fully connected (FC) layer is applied to produce a style code as a vector consisting of 8 digits, in which, 2 final digits represent the information code (disentangled style) $I_i$ of the image.

#### 4.1.3. Generator

The generator takes content code and style code as inputs to reconstruct the initial input image. The content code goes through residual blocks and upsampling layers. These residual blocks are upgraded with Adaptive Instance Normalization (AdaIn) layers [37] which receive style parameters from a multilayer perception (MLP) which has the style code as its input.

#### 4.1.4. Multi-purpose Discriminator

Our discriminator consists of two branches. The first branch is a traditional discriminator which can be found in most of the GAN-based models. The second branch consists of convolutional blocks to learn the Q distribution. These two branches share the first convolutional blocks.

#### 4.1.5. Hyperparameters

In all our experiments in the paper, we apply Adam optimizer with $\beta_1$ and $\beta_2$ as 0.5 and 0.999 respectively. The learning rate is initially set to 0.0001, with a weight decay of 0.0001 applied every 100 thousand iterations. Our weight losses are $\lambda_x = 10$, $\lambda_c = \lambda_s = \lambda_{mi} = 1$.

#### 4.1.6. Baselines

We compare our proposed method InfoMUNIT with following unpaired image-to-image translation techniques: CycleGAN[19], MUNIT[9] and DRIT++[22]. The training procedures of those methods are done using official source code and configurations provided by their authors on GitHub.com.

### 4.2. Evaluation

We use three performance measures that estimate the quality and the diversity of the generated images, to compare InfoMUNIT with the baselines.

#### 4.2.1. Conditional Inception Score

Based on Inception Score (IS) [38], Huang et al. [9] introduced Conditional Inception Score (CIS) specified for evaluation of multimodal image-to-image tasks. While IS measures the quality and diversity of all generated images at once, CIS focuses on the diversity of images that are translated from the same input image. Having multiple input images in the test set, we compute CIS for each group of images generated from the same input, and finally, take the mean CIS for the whole test set.

#### 4.2.2. Frechet Inception Distance

Frechet Inception Distance (FID) [39] computes the distance between the set of generated images and the set images in the target domain. It is computed by calculated the distance between the Inception feature vectors for the two sets of images. Thus, FID can be used for evaluating networks that are trained on specific datasets without requiring a classifier pretrained on an alike dataset. The lower FID we have, the more realistic the generated images are. Normally, those feature vectors are taken from the third pooling layer of the Inception model which contains 2048 features. Due to the small size of our datasets, we compute the distance using features of the second pooling layer containing 192 features.

#### 4.2.3. LPIPS Distance

The translation diversity is also measured by LPIPS distance which is shown in [40] to be highly correlated with human judgment. We compute LPIPS distance on generated samples of each input image, then take the average value. The larger distance among them, the more diverse they are.

### 4.3. Datasets

We use multiple datasets for evaluating InfoMUNIT and compare its performance with state-of-the-art techniques on the task of image-to-image translation. Each dataset contains two sets of images and our network is trained to transform images between the two domains. We crop and down-sample all images to the size of $64 \times 64$, in RGB-color mode.

#### 4.3.1. Edges↔Shoes and Edges↔Bags

These two datasets contain images of shoes and handbags along with their edges, introduced in the work of Isola et al. [12]. The edges↔shoes dataset contains 138667 pairs of samples while the edges↔bags dataset contains 49925 pairs. From each dataset, we keep 200 pairs of samples for testing and the rest for training. Note that we do not use the paired information of these two datasets.

#### 4.3.2. Cats↔Dogs

The dataset is comprised of 1364 photos of dogs and 871 photos of cats, cropped to their heads [22]. We keep 100 images from each set for testing while the rest is used for training.

#### 4.3.3. Portraits (Painted↔Real)

This dataset consists of 1814 painted portraits and real 6452 portraits captured by cameras [22]. We keep 100 images from each set for testing while using the rest for training.

## 5. Results

### 5.1. Image Quality

The qualitative comparison of InfoMUNIT and other methods is shown in Figure 2. The objective of InfoMUNIT is to increase the diversity and ability to control features of generated images compared to MUNIT and the state-of-the-art, while not hurting their quality. As being shown in Figure 2, the quality of images generated by InfoMUNIT are at least as good as the images from other methods. The result is confirmed in Table 1 where we apply FID to quantitatively evaluate the realism of the generated images. Even


Figure 2: Random samples generated by our method and baselines, trained on two datasets: edges→bags (left) and cats→dogs (right). The input images (and ground-truths) are displayed in the first column. Other columns show random outputs of baseline methods and InfoMUNIT.

Table 1: Frechet Inception Distance (FID). Lower value means better performance.

| | **InfoMUNIT** | **MUNIT** | **CycleGAN** | **DRIT++** |
|---|---|---|---|---|
| edge2bag | 2.81 | 2.56 | 4.23 | **1.69** |
| bag2edge | 7.68 | 8.52 | 58.53 | **5.64** |
| edge2shoe | 1.28 | 1.44 | 4.86 | **1.13** |
| shoe2edge | 4.69 | 8.83 | 88.03 | **4.24** |
| dog2cat | 9.24 | 13.48 | **2.56** | 21.59 |
| cat2dog | 6.31 | 6.31 | **2.02** | 18.14 |
| paint2real | 2.96 | 3.02 | **2.56** | 7.29 |
| real2paint | 8.60 | 8.51 | **3.97** | 18.85 |
| Average | **5.45** | 6.58 | 20.84 | 9.82 |

though InfoMUNIT does not out outperform other methods in terms of image quality in any task, its performance is stable across all tasks. The performance of InfoMUNIT is close to the best method for each datatset. InfoMUNIT gives performance equivalent or better than MUNIT. This shows that the disentangled features have also an impact on the quality of the images. Notice that DRIT++ is the best for the first four tasks but totally fails in the last four tasks. This is illustrated by the strange dog images generated by DRIT++ in the Figure 2. On the opposite, CycleGAN gives the best performance for the last four tasks but is bad in the first four tasks and especially in the bag2edge and shoe2edge tasks. On average, InfoMUNIT achieves the best FID value among the four image-to-image translation methods. The good quality of images generated by InfoMUNIT is stable on multiple datasets.

### 5.2. Image Diversity

Table 2 and Table 3 respectively shows the CIS and LPIPS scores that evaluate the diversity of generated images. CycleGAN is not a multimodal method, it can generate only one output from one input so it does therefore not appear in this table. The LPIPS and CIS scores of InfoMUNIT are clearly superior to the scores of DRIT++ and MUNIT. The only exceptions are for the shoe2edge task where the LPIPS of DRIT++ is higher and for the real2paint task where the LPIPS of MUNIT is higher. In both cases the LPIPS of InfoMUNIT is very close to the best score and still higher than the LPIPS of the third

Table 2: LPIPS distance. Higher value means better performance.

| | **InfoMUNIT** | **MUNIT** | **DRIT++** |
|---|---|---|---|
| edge2bag | **3.00** | 2.07 | 2.13 |
| bag2edge | **2.01** | 1.07 | 1.60 |
| edge2shoe | **2.35** | 2.23 | 1.76 |
| shoe2edge | 1.44 | 1.00 | **1.51** |
| dog2cat | **2.24** | 1.97 | 1.11 |
| cat2dog | **2.65** | 2.24 | 1.09 |
| paint2real | **1.96** | 1.91 | 1.74 |
| real2paint | 2.14 | **2.26** | 1.14 |
| Average | **2.40** | 1.88 | 1.51 |

Table 3: Conditional Inception Score (CIS). Higher value means better performance.

| | **InfoMUNIT** | **MUNIT** | **DRIT++** |
|---|---|---|---|
| edge2bag | **0.42** | 0.29 | 0.30 |
| bag2edge | **0.35** | 0.04 | 0.22 |
| edge2shoe | **0.26** | 0.24 | 0.22 |
| shoe2edge | **0.24** | 0.00 | 0.12 |
| dog2cat | **0.32** | **0.32** | 0.04 |
| cat2dog | **0.30** | 0.28 | 0.03 |
| paint2real | **0.25** | **0.25** | 0.11 |
| real2paint | **0.33** | 0.30 | 0.06 |
| Average | **0.31** | 0.21 | 0.14 |

method. Over all datasets, the scores of InfoMUNIT are significantly better than the other methods. Figure 3 and Figure 4 illustrate the higher diversity of InfoMUNIT compared to MUNIT. This results show that InfoMUNIT generates significantly more diverse outputs than MUNIT and DRIT++.

### 5.3. Controlling Features

In this subsection, we show the advantage of InfoMUNIT over its predecessor MUNIT in manipulating features. From Figure 3, we can observe that varying values of style code in MUNIT can lead to slight changes like color of the object. With InfoMUNIT, we can significantly manipulate the features of the object. The first disentangled feature controls the size of the bag and the second one control the color from white to black. We also notice that InfoMUNIT is able to propose different textures of the bag.

The performance of InfoMUNIT on the edges→shoes task is illustrated in Figure 4 and Figure 5. While MUNIT can only change some small details of the shoes, we can significantly manipulate the color of the shoes with InfoMUNIT. Varying the first info style code makes the color changed from bright to dark, while varying the second one changes the color from cold to warm. In Figure 4, we can see that the first info style code is also responsible for the style of the shoes. From the left to the right, it turns a sneaker to a pump and makes it darker at the same time. This effect makes sense as pumps are more likely to have dark colors than sneakers.

Please note that the value of each disentangled feature in this test is plotted from $-2$ to 2 instead of $-1$ to 1 in the training phase, which means the generator is receiving style code values that it has never seen before. This explains why the images on the border looks a

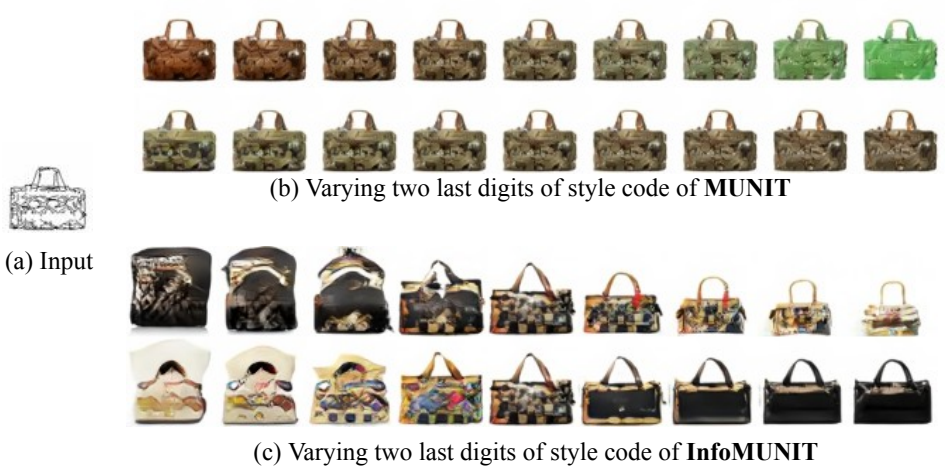



Figure 3: Manipulating two last digits in the style code of MUNIT and InfoMUNIT on edges→bags task.


Figure 4: Manipulating two last digits in the style code of MUNIT and InfoMUNIT on edges→shoes task.

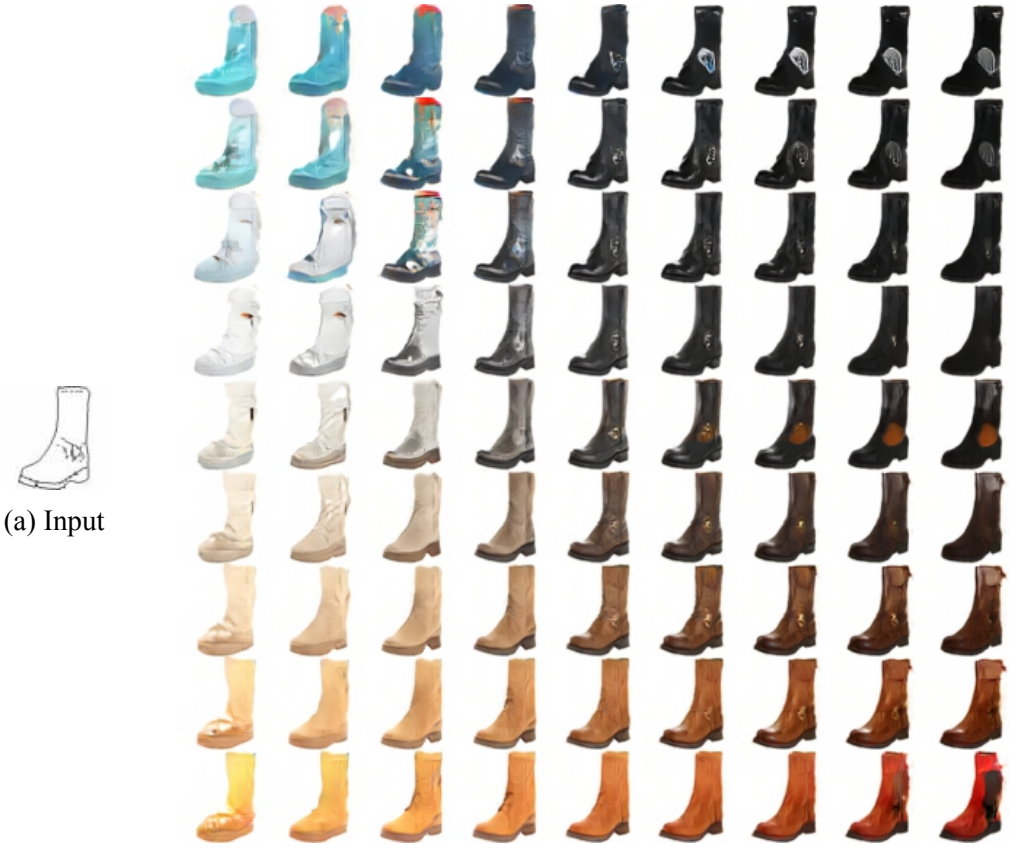
bit extreme.

### 5.4. The length of information latent code

We perform some experiments to investigate the impact of the length of information latent code $i$ on the generated images in varying this value from 1 to 8. Table 4 shows some of these results on the *edge2shoe* datasets. We see that the FID, CIS and LPIPS weakly vary with the length of $i$. We conclude from these results that the quality and diversity of the generated images by InfoMUNIT are robust to the length of the information latent code.

## 6. Conclusion

We proposed an extension of MUNIT called InfoMUNIT which can manipulate features of the translated images. Our method is demonstrated in multiple image-to-image translation tasks. It achieves comparable translated image quality to state-of-the-art approaches and outperforms them in terms of outputs diversity. Moreover, our method improves the control of the user on the generated images, this kind of tool can make the image manipulation method more usable for real life applications.



(a) Input

(b) Outputs generated by **InfoMUNIT**

Figure 5: Combination of the two last digits in the style code of InfoMUNIT on edges→shoes task. From left to right (b), we vary the value of the first information latent code. From top to bottom, we vary the second one.

Table 4: Performance of InfoMUNIT with different lengths of information latent code.

| **Length of i** | **1** | **2** | **4** | **6** | **8** |
|---|---|---|---|---|---|
| **FID** | 2.99 | **2.81** | 3.19 | 3.11 | 3.37 |
| **CIS** | 2.59 | 3.00 | 3.64 | 3.61 | **3.65** |
| **LPIPS** | 0.42 | 0.42 | **0.47** | **0.47** | 0.46 |

## References


[1] S. Iizuka, E. Simo-Serra, and H. Ishikawa, "Let there be color! joint end-to-end learning of global and local image priors for automatic image colorization with simultaneous classification," *ACM Transactions on Graphics (ToG)*, vol. 35, no. 4, pp. 1–11, 2016.

[2] L. A. Gatys, A. S. Ecker, and M. Bethge, "Image style transfer using convolutional neural networks," in *Proceedings of the IEEE conference on computer vision and pattern recognition*, pp. 2414–2423, 2016.

[3] C. Dong, C. C. Loy, K. He, and X. Tang, "Learning a deep convolutional network for image super-resolution," in *European conference on computer vision*, pp. 184–199, Springer, 2014.

[4] J. Xie, L. Xu, and E. Chen, "Image denoising and inpainting with deep neural networks," in *Advances in neural information processing systems*, pp. 341–349, 2012.

[5] Z. Murez, S. Kolouri, D. Kriegman, R. Ramamoorthi, and K. Kim, "Image to image translation for domain adaptation," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pp. 4500–4509, 2018.

[6] M.-Y. Liu, T. Breuel, and J. Kautz, "Unsupervised image-to-image translation networks," in *Advances in neural information processing systems*, pp. 700–708, 2017.

[7] J. Hoffman, E. Tzeng, T. Park, J.-Y. Zhu, P. Isola, K. Saenko, A. A. Efros, and T. Darrell, "Cycada: Cycle-consistent adversarial domain adaptation," *arXiv preprint arXiv:1711.03213*, 2017.

[8] H.-Y. Lee, H.-Y. Tseng, J.-B. Huang, M. Singh, and M.-H. Yang, "Diverse image-to-image translation via disentangled representations," in *Proceedings of the European conference on computer vision (ECCV)*, pp. 35–51, 2018.

[9] X. Huang, M.-Y. Liu, S. Belongie, and J. Kautz, "Multimodal unsupervised image-to-image translation," in *Proceedings of the European Conference on Computer Vision (ECCV)*, pp. 172–189, 2018.

[10] J. P. Cohen, M. Luck, and S. Honari, "Distribution matching losses can hallucinate features in medical image translation," in *International conference on medical image computing and computer-assisted intervention*, pp. 529–536, Springer, 2018.

[11] I. Goodfellow, J. Pouget-Abadie, M. Mirza, B. Xu, D. Warde-Farley, S. Ozair, A. Courville, and Y. Bengio, "Generative adversarial nets," in *Advances in neural information processing systems*, pp. 2672–2680, 2014.

[12] P. Isola, J.-Y. Zhu, T. Zhou, and A. A. Efros, "Image-to-image translation with conditional adversarial networks," in *Proceedings of the IEEE conference on computer vision and pattern recognition*, pp. 1125–1134, 2017.

[13] T.-C. Wang, M.-Y. Liu, J.-Y. Zhu, A. Tao, J. Kautz, and B. Catanzaro, "High-resolution image synthesis and semantic manipulation with conditional gans," in *Proceedings of the IEEE conference on computer vision and pattern recognition*, pp. 8798–8807, 2018.

[14] A. Shrivastava, T. Pfister, O. Tuzel, J. Susskind, W. Wang, and R. Webb, "Learning from simulated and unsupervised images through adversarial training," in *Proceedings of the IEEE conference on computer vision and pattern recognition*, pp. 2107–2116, 2017.

[15] K. Bousmalis, N. Silberman, D. Dohan, D. Erhan, and D. Krishnan, "Unsupervised pixel-level domain adaptation with generative adversarial networks," in *Proceedings of the IEEE conference on computer vision and pattern recognition*, pp. 3722–3731, 2017.

[16] Y. Taigman, A. Polyak, and L. Wolf, "Unsupervised cross-domain image generation," *arXiv preprint arXiv:1611.02200*, 2016.

[17] S. Benaim and L. Wolf, "One-sided unsupervised domain mapping," in *Advances in neural information processing systems*, pp. 752–762, 2017.

[18] Z. Yi, H. Zhang, P. Tan, and M. Gong, "Dualgan: Unsupervised dual learning for image-to-image translation," in *Proceedings of the IEEE international conference on computer vision*, pp. 2849–2857, 2017.

[19] J.-Y. Zhu, T. Park, P. Isola, and A. A. Efros, "Unpaired image-to-image translation using cycle-consistent adversarial networks," in *Proceedings of the IEEE international conference on computer vision*, pp. 2223–2232, 2017.

[20] T. Kim, M. Cha, H. Kim, J. K. Lee, and J. Kim, "Learning to discover cross-domain relations with generative adversarial networks," in *Proceedings of the 34th International Conference on Machine Learning-Volume 70*, pp. 1857–1865, JMLR. org, 2017.

[21] A. Almahairi, S. Rajeswar, A. Sordoni, P. Bachman, and A. Courville, "Augmented cyclegan: Learning many-to-many mappings from unpaired data," *arXiv preprint arXiv:1802.10151*, 2018.

[22] H.-Y. Lee, H.-Y. Tseng, Q. Mao, J.-B. Huang, Y.-D. Lu, M. Singh, and M.-H. Yang, "Drit++: Diverse image-to-image translation via disentangled representations," *International Journal of Computer Vision*, pp. 1–16, 2020.

[23] G. E. Hinton, S. Osindero, and Y.-W. Teh, "A fast learning algorithm for deep belief nets," *Neural computation*, vol. 18, no. 7, pp. 1527–1554, 2006.

[24] P. Vincent, H. Larochelle, Y. Bengio, and P.-A. Manzagol, "Extracting and composing robust features with denoising autoencoders," in *Proceedings of the 25th international conference on Machine learning*, pp. 1096–1103, 2008.

[25] A. Rasmus, M. Berglund, M. Honkala, H. Valpola, and T. Raiko, "Semi-supervised learning with ladder networks," in *Advances in neural information processing systems*, pp. 3546–3554, 2015.

[26] L. Maaløe, C. K. Sønderby, S. K. Sønderby, and O. Winther, "Improving semi-supervised learning with auxiliary deep generative models," in *NIPS Workshop on Advances in Approximate Bayesian Inference*, 2015.

[27] A. Radford, L. Metz, and S. Chintala, "Unsupervised representation learning with deep convolutional generative adversarial networks," *arXiv preprint arXiv:1511.06434*, 2015.

[28] J. B. Tenenbaum and W. T. Freeman, "Separating style and content with bilinear models," *Neural computation*, vol. 12, no. 6, pp. 1247–1283, 2000.

[29] Z. Zhu, P. Luo, X. Wang, and X. Tang, "Multi-view perceptron: a deep model for learning face identity and view representations," in *Advances in Neural Information Processing Systems*, pp. 217–225, 2014.

[30] D. P. Kingma, S. Mohamed, D. J. Rezende, and M. Welling, "Semi-supervised learning with deep generative models," in *Advances in neural information processing systems*, pp. 3581–3589, 2014.

[31] A. Makhzani, J. Shlens, N. Jaitly, I. Goodfellow, and B. Frey, "Adversarial autoencoders," *arXiv preprint arXiv:1511.05644*, 2015.

[32] D. Barber and F. V. Agakov, "Kernelized infomax clustering," in *Advances in neural information processing systems*, pp. 17–24, 2006.

[33] T. D. Kulkarni, W. F. Whitney, P. Kohli, and J. Tenenbaum, "Deep convolutional inverse graphics network," in *Advances in neural information processing systems*, pp. 2539–2547, 2015.

[34] G. Desjardins, A. Courville, and Y. Bengio, "Disentangling factors of variation via generative entangling," *arXiv preprint arXiv:1210.5474*, 2012.

[35] X. Chen, Y. Duan, R. Houthooft, J. Schulman, I. Sutskever, and P. Abbeel, "Infogan: Interpretable representation learning by information maximizing generative adversarial nets," in *Advances in neural information processing systems*, pp. 2172–2180, 2016.

[36] D. Barber and F. V. Agakov, "The im algorithm: a variational approach to information maximization," in *Advances in neural information processing systems*, p. None, 2003.

[37] X. Huang and S. Belongie, "Arbitrary style transfer in real-time with adaptive instance normalization," in *Proceedings of the IEEE International Conference on Computer Vision*, pp. 1501–1510, 2017.

[38] T. Salimans, I. Goodfellow, W. Zaremba, V. Cheung, A. Radford, and X. Chen, "Improved techniques for training gans," in *Advances in neural information processing systems*, pp. 2234–2242, 2016.

[39] M. Heusel, H. Ramsauer, T. Unterthiner, B. Nessler, and S. Hochreiter, "Gans trained by a two time-scale update rule converge to a local nash equilibrium," in *Advances in neural information processing systems*, pp. 6626–6637, 2017.

[40] R. Zhang, P. Isola, A. A. Efros, E. Shechtman, and O. Wang, "The unreasonable effectiveness of deep features as a perceptual metric," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pp. 586–595, 2018.

université PARIS-SACLAY

ÉCOLE DOCTORALE
Sciences et technologies de l'information et de la communication (STIC)

**Titre:** Réalisation d'une Aide au Diagnostic en Orthodontie par Apprentissage Profond

**Mots clés:** Orthodontie; Diagnostic; Apprentissage profond; Adaptation de Domaine; Transformation d'Image; Few-Shot Learning

**Résumé:** L'analyse et le diagnostic précis à partir d'images dentaires sont un facteur essentiel de la réussite des traitements orthodontiques. De nombreux procédés de traitement d'image ont été proposés pour résoudre ce problème. Cependant, ces études fonctionnent principalement sur de petits ensembles de données de radiographies dans des conditions de laboratoire et ne sont pas vraiment applicables en tant que produits ou services complets. Dans cette thèse, nous construisons des modèles d'apprentissage profond pour diagnostiquer des problèmes dentaires tels que la gingivite et les dents chevauchées à l'aide de photos prises par de téléphones portables. Nous étudions les couches cachées de ces modèles pour trouver les forces et les limites de chaque méthode. Nous proposons un pipeline complet intégrant le prétraitement des images, l'apprentissage du modèle et le post-traitement des résultats pour créer un processus d'analyse complet prêt à être mis en production en situation réel. Afin d'améliorer la fiabilité des modèles, nous avons étudié différentes méthodes d'augmentation des données, en particulier les méthodes d'adaptation de domaine en utilisant des approche de transfert d'images, à la fois supervisée et non supervisée, et obtenons des résultats prometteurs. Les approches de transformation d'images sont également utilisés pour simplifier le choix des appareils orthodontiques par les patients en leur montrant à quoi pourraient ressembler leurs dents pendant le traitement. Nos méthodes permettent de générées des images réalistes et en haute définition. Nous proposons également un nouveau modèle de transformation d'image non supervisé qui peut manipuler les caractéristiques de l'image sans nécessiter d'annotation supplémentaire. Notre modèle surpasse les techniques de pointe sur plusieurs applications de transformation d'images et est également étendu pour les problèmes de « few-shot learning ».

**Title:** Realization of a Diagnostic Aid in Orthodontics by Deep Learning

**Keywords:** Orthodontics; Diagnosis; Deep Learning; Domain Adaptation; Image-to-Image Translation; Few-Shot Learning

**Abstract:** Accurate processing and diagnosis of dental images is an essential factor determining the success of orthodontic treatment. Many image processing methods have been proposed to address this problem. Those studies mainly work on small datasets of radiographs under laboratory conditions and are not highly applicable as complete products or services. In this thesis, we train deep learning models to diagnose dental problems such as gingivitis and crowded teeth using mobile phones' images. We study feature layers of these models to find the strengths and limitations of each method. Besides training deep learning models, we also embed each of them in a pipeline, including preprocessing and post-processing steps, to create a complete product. For the lack of training data problem, we studied a variety of methods for data augmentation, especially domain adaptation methods using image-to-image translation models, both supervised and unsupervised, and obtain promising results. Image translation networks are also used to simplifying patients' choice of orthodontic appliances by showing them how their teeth could look like during treatment. Generated images have are realistic and in high resolution. Researching further into unsupervised image translation neural networks, we propose an unsupervised image-to-image translation model which can manipulate features of objects in the image without requiring additional annotation. Our model outperforms state-of-the-art techniques on multiple image translation applications and is also extended for few-shot learning problems.

Université Paris-Saclay
Espace Technologique / Immeuble Discovery
Route de l'Orme aux Merisiers RD 128 / 91190 Saint-Aubin, France